Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Ataque de Negação de Serviço por Reflexão Amplificada usando Simple Service Discovery Protocol

Henrique Senoo Hirata

Monografia apresentada como requisito parcial
para conclusão do Curso de Engenharia da Computação

Orientador
Prof. Dr. João José Costa Gondim

Brasília
2018

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Ataque de Negação de Serviço por Reflexão Amplificada usando Simple Service Discovery Protocol

Henrique Senoo Hirata

Monografia apresentada como requisito parcial
para conclusão do Curso de Engenharia da Computação

Prof. Dr. João José Costa Gondim (Orientador)
CIC/UnB

Prof. Dr. Marcos Fagundes Caetano
CIC/UnB

Prof. Dr. Marcelo Grandi Mandelli
CIC/UnB

Prof. Dr. José Edil Guimarães de Medeiros
Coordenador do Curso de Engenharia da Computação

Brasília, 29 de novembro de 2018

Dedicatória

Dedico este trabalho à minha família e a toda a comunidade interessada na segurança cibernética.

Agradecimentos

Agradeço à minha família por todo apoio ao longo do período acadêmico.

Agradeço aos meus amigos que me apoiaram em momentos difíceis e me ajudaram a superar obstáculos na graduação.

Agradeço aos meus professores, que me passaram conhecimentos valiosos para a minha formação.

Agradeço ao Departamento de Ciência da Computação por todo o apoio técnico ao longo do curso.

Agradeço ao meu orientador João José Costa Gondim pelo apoio nesta fase final do curso, o que possibilitou a realização deste trabalho.

Resumo

Com a popularização de dispositivos IoT, há uma grande preocupação com a segurança cibernética. Uma das preocupações é com ataques de negação de serviço. Apesar de serem usualmente associados a hacktivismo na mídia de massa, esses ataques podem gerar grandes prejuízos e até acobertar outros ataques. Pensando nisto, este trabalho apresenta um estudo do ataque de negação de serviço por reflexão amplificada usando o método M-SEARCH do protocolo *Simple Service Discovery Protocol* (SSDP), um protocolo usado pelo *Universal Plug and Play* (UPnP). O UPnP por sua vez é um conjunto de protocolos usado para conectar dispositivos sem a necessidade de configuração por parte do usuário. Para auxílio do estudo, foi implementada uma ferramenta que busca por refletores e realiza o ataque. Esta ferramenta se trata de uma evolução daquela desenvolvida pelo ex-aluno Tiago Fonseca Medeiros em seu projeto final de graduação [1]. Além de ser alterada para suportar o protocolo SSDP, a ferramenta foi modificada de forma a torná-la mais fácil de estender com novos protocolos. Nos testes, a ferramenta conseguiu realizar ataques amplificados na ordem de 600 vezes para o protocolo SNMP e 38 vezes para o SSDP e permitiu verificar o comportamento de saturação do refletor.

Palavras-chave: DoS, SSDP, M-SEARCH, UPnP, Reflexão Amplificada

Abstract

As the number of IoT devices rises, a great concern with Cybersecurity arises. Denial of Service Attacks are among them. While usually associated with hacktivism in mass media, theses attacks can incur in big financial losses and even cover other attacks. Those are the motivations of this study on Reflective Denial of Service Attacks, abusing *Simple Service Discovery Protocol* M-SEARCH messages. SSDP is a protocol used by the *Universal Plug and Play* (UPnP), which by its turn is a protocol stack used to connect devices without extra configuration input by the end user. To aid the study, a tool that search for reflectors and performs the attack was implemented. This tool is an evolution of another tool developed in [1]. The tool was modified to accept SSDP and to make it easier to implement new protocols. In tests, the tool managed to perform amplified attacks with a factor 600 with SNMP and factor 38 with SSDP. It also demonstrated saturation in reflector behavior.

Keywords: DoS, SSDP, M-SEARCH, UPnP, Reflexão Amplificada

Sumário

Sumário vii

Lista de Figuras ix

Lista de Tabelas x

Lista de Abreviaturas e Siglas xi

1 Introdução 1

1.1 Justificativa 1

1.2 Objetivos 2

1.3 Organização do Trabalho 3

2 Revisão Conceitual 4

2.1 Ataques de Negação de Serviço 4

2.1.1 Ataques Volumétricos 4

2.1.2 Ataques por Reflexão Amplificada 4

2.1.3 IP Spoofing 5

2.1.4 Ataques da Camada de Aplicação 7

2.1.5 Ataques *Low and Slow* 7

2.1.6 Ataques Distribuídos de Negação de Serviço 7

2.2 O Conjunto de Protocolos UPnP 8

2.2.1 Distribuição de Versões 8

2.2.2 UPnP Device Architecture 1.0 (UDA 1.0) 9

2.3 O Ataque com Protocolo SSDP 16

2.3.1 Potencial de Amplificação 17

2.3.2 Balanceamento de Carga 17

2.3.3 Reflexão 17

2.3.4 Metodologia do Ataque 18

2.4 O Ataque com Protocolo SNMP 18
2.4.1 Reflexão 18
2.4.2 Amplificação 19
2.4.3 Balanceamento de Carga 19
2.4.4 Metodologia do Ataque 19
2.5 Mitigação dos Ataques 20
2.5.1 *Spoofing* de IP 20
2.5.2 Vulnerabilidades dos Protocolos 20
2.6 Considerações Finais 21

3 A Ferramenta **22**
3.1 Arquitetura 22
3.1.1 GUI – Módulo de Interface do Usuário 23
3.1.2 Scanner – Módulo de Escaneamento 26
3.1.3 Striker – Módulo de Ataque 26
3.2 Alterações da Ferramenta Original 27
3.2.1 Alterações na Implementação da GUI 27
3.2.2 Alterações na Implementação do Scanner 28
3.2.3 Alterações na Implementação da Importação e Exportação 29
3.3 Implementação do Módulo SSDP 29
3.4 Comparativo com Outras Ferramentas 30
3.5 Considerações Finais 30

4 Testes e Resultados **31**
4.1 Análise de Desempenho 32
4.1.1 Atacante e Refletor 34
4.1.2 Refletor e Alvo 36
4.2 Análise da Amplificação 39
4.3 Teste com SSDP e SNMP simultaneamente 41
4.4 Considerações Finais 42

5 Conclusão **43**
5.1 Trabalhos Futuros 44

Referências **45**

Lista de Figuras

2.1 *Spoofing* em TCP com uma conexão previamente estabelecida. 5
2.2 Estabelecimento de uma conexão TCP. 6
2.3 Estrutura lógica de um dispositivo UPnP. 9
2.4 Passos do UPnP. 10

3.1 Módulos da ferramenta. 22
3.2 A janela principal da ferramenta. 23
3.3 Janela com a lista de refletores. 24
3.4 Janela de escaneamento com SNMP selecionado. 25
3.5 Janela de escaneamento com SSDP selecionado. 26

4.1 Fluxo de dados entre atacante e refletor SNMP. 34
4.2 Fluxo de pacotes entre atacante e refletor SNMP. 35
4.3 Fluxo de dados entre atacante e refletor SSDP. 35
4.4 Fluxo de pacotes entre atacante e refletor SSDP. 36
4.5 Fluxo de dados entre refletor e alvo SNMP. 37
4.6 Fluxo de pacotes entre refletor e alvo SNMP. 37
4.7 Fluxo de dados entre refletor e alvo SSDP. 38
4.8 Fluxo de pacotes entre refletor e alvo SSDP. 38
4.9 Amplificação em bits com protocolo SNMP. 39
4.10 Amplificação em pacotes com protocolo SNMP. 40
4.11 Amplificação em bits com protocolo SSDP. 40
4.12 Amplificação em pacotes com protocolo SSDP. 41

Lista de Tabelas

2.1 Descrição dos campos da mensagem NOTIFY. 13
2.2 Descrição dos campos da mensagem M-SEARCH. 14
2.3 Descrição dos campos da mensagem de resposta ao M-SEARCH. 15

3.1 Opções de escaneamento para o SNMP. 25
3.2 Opções de escaneamento para o SSDP. 25

4.1 SNMP – Fluxo de dados em bits por segundo. 33
4.2 SNMP – Fluxo de dados em pacotes por segundo. 33
4.3 SSDP – Fluxo de dados em bits por segundo. 33
4.4 SSDP – Fluxo de dados em pacotes por segundo. 33
4.5 Teste com SSDP e SNMP simultaneamente. 41

Lista de Abreviaturas e Siglas

ACK *Acknowledgment.*

API *Application Programming Interface.*

DDoS *Distributed Denial of Service.*

DHCP *Dynamic Host Configuration Protocol.*

DNS *Domain Name System.*

DoS *Denial of Service.*

GENA *General Event Notification Architecture.*

GUI *Graphical User Interface.*

HTML *HyperText Markup Language.*

HTTP *Hypertext Transfer Protocol.*

IGDv2 *Internet Gateway Device version 2.*

IP *Internet Protocol.*

IPv4 *Internet Protocol version 4.*

JSON *JavaScript Object Notation.*

LOIC *Low Orbit Ion Cannon.*

MIB *Management Information Base.*

NTP *Network Time Protocol.*

PnP *Plug and Play.*

SDK *Software Development Kit.*

SNMP *Simple Network Management Protocol.*

SOAP *Simple Object Access Protocol.*

SSDP *Simple Service Discovery Protocol.*

SYN *Synchronize.*

TCP *Transmission Control Protocol.*

UDA *UPnP Device Architecture.*

UDP *User Datagram Protocol.*

UPnP *Universal Plug and Play.*

WAN *Wide Area Network.*

XML *Extensible Markup Language.*

zeroconf *zero-configuration networking.*

Capítulo 1

Introdução

Os ataques de *Denial of Service* (DoS), negação de serviço em inglês, vem se tornando cada vez mais populares através dos anos. O relatório da Verisign do primeiro quarto de 2018 [2] mostra um aumento de 53% no número de ataques e 47% no tamanho médio nos ápices dos ataques em relação ao ano passado. A atratividade desses ataques se deve as características básicas de um ataque por reflexão amplificada, o tipo mais comum conforme mostra [3]. Esses ataques necessitam de menos recursos, graças a amplificação, e são mais difíceis de mitigar, já que é difícil diferenciar o trafego legítimo com o trafego de vários refletores diferentes. Outro ponto chave é que não é necessária qualquer invasão ou quebra de segurança do alvo, a vulnerabilidade em si está nos refletores. Outro ponto é o IP *spoofing*, um problema que poderia ser mitigado caso os provedores de internet seguissem o BCP 38[1]. Entretanto, tal prática gera maior custo de processamento sem nenhuma contrapartida direta para os provedores, resultando na baixa adoção dessa prática.

1.1 Justificativa

Conforme dito anteriormente, o alvo não é a priori o possibilitador do ataque, entretanto, não se deve depender de fatores externos para executar a segurança própria.

Os prejuízos causados são enormes. De acordo com [4], foram gastos, por ataque, em média mais de 120 mil dólares pelas pequenas e médias empresas e 2 milhões pelas grandes. Dentre os ataques notáveis que afetaram grandes empresas, podem-se citar o ataque realizado pelo grupo *Lizard Squad* no natal de 2014 [5], que interrompeu os serviços da *PlayStation Network* e *Xbox Live*, e o ataque ao provedor DNS Dyn [6], que interrompeu o serviço de grandes sites tais como Paypal, Netflix e Twitter.

Além de prejuízos, ataques DoS podem ser usados em um contexto geopolítico. Nestes casos, ataques podem ser feitos para afetar estruturas governamentais ou até mesmo como

[1] `https://tools.ietf.org/html/bcp38`

propaganda para certas ideologias. Um exemplo de um ataque com estas motivações foi o ataque ao GitHub em 2015 [7], em que projetos destinados a combater a censura na China foram alvos de ataques aparentemente originados de dentro desse país.

Outro ponto crítico é o aumento de dispositivos conectados a cada ano. Segundo [8], em 2015 haviam mais de 4,9 bilhões de dispositivos IoT em uso e uma previsão de mais de 20,7 bilhões de dispositivos para 2020. Um maior número de dispositivos pode significar um aumento da capacidade dos ataques DoS, caso não haja a proteção adequada desses dispositivos.

Dados estas consequências, é necessário estudar esses ataques para mitigar seus efeitos e detectar dispositivos que possam ser usados como refletores.

1.2 Objetivos

O primeiro objetivo deste trabalho é realizar um estudo do protocolo *Simple Service Discovery Protocol* (SSDP) como vetor de ataque de negação de serviço por reflexão amplificada. Este estudo busca esclarecer o funcionamento do ataque e possíveis maneiras de mitigá-lo. Reportado pela primeira vez em 2014, o ataque de negação de serviço com SSDP foi escolhido devido a sua atratividade para os atacantes. Isto se deve a grande quantidade de dispositivos vulneráveis, conforme pode ser visto em [9] e [10], o que possibilita ataques mais efetivos.

O segundo objetivo é aprimorar a ferramenta desenvolvida em [1]. Originalmente realizando o ataque por abuso do protocolo *Simple Network Management Protocol* (SNMP), a ferramenta faz detecção de refletores com calculo da amplificação máxima. Além disso, a ferramenta pode realizar ataques com base nos dados previamente obtidos pela detecção. O objetivo neste trabalho é implementar o uso do protocolo SSDP, um protocolo usado pelo *Universal Plug and Play* (UPnP), e modificar a arquitetura da ferramenta de modo que outros protocolos possam ser implementados mais facilmente.

A fim de atestar a continuidade do funcionamento e da viabilidade do ataque com SNMP na nova implementação, foram realizados novos testes com este protocolo. O diferencial do teste realizado aqui é o uso de outra implementação SNMP, o que pode prover novos dados sobre este ataque.

Esta ferramenta está sendo desenvolvida com fins didáticos e acadêmicos, para fins de demonstração de vulnerabilidades visando o desenvolvimento de melhorias na mitigação de ataques. O código da mesma se encontra sob guarda do autor e do orientador deste trabalho.

1.3 Organização do Trabalho

O Capítulo 2 aborda os conceitos teóricos em que o ataque se baseia. No Capítulo 3 a ferramenta é apresentada, com abordagem do design, da implementação e do uso da ferramenta. O Capítulo 4 contêm os resultados obtidos dos testes realizados com a ferramenta. Finalmente o Capítulo 5 apresenta as conclusões e sugestões de trabalhos futuros.

Capítulo 2

Revisão Conceitual

Neste capítulo são apresentados os conceitos que formam a base do ataque. Na primeira parte são explicados os vários tipos de ataques de negação de serviço. Em seguida é apresentado uma breve explicação do UPnP, com ênfase na parte de descobrimento de dispositivos. Finalmente são abordados os ataques SSDP e SNMP.

2.1 Ataques de Negação de Serviço

Diferentemente de ataques tradicionais, os ataques DoS não incorrem em uma quebra de segurança, invasão ou violação de dados, o objetivo é impedir o acesso dos usuários legítimos a um serviço.

2.1.1 Ataques Volumétricos

Os ataques volumétricos são uma abordagem por força bruta, onde um grande volume de trafego é enviado para um alvo de forma a sobrecarregar sua infraestrutura. Como dificilmente apenas a infraestrutura do atacante é suficiente para realizar tal feito, as técnicas de ataques distribuídos e de reflexão amplificada podem ser utilizadas. Estes ataques são explicados nas próximas seções.

2.1.2 Ataques por Reflexão Amplificada

O ataque de reflexão amplificada é um ataque volumétrico no qual o atacante se aproveita de protocolos que enviam respostas que são muito maiores que a requisição. Conhecido um protocolo com esta característica, o atacante então deve mandar requisições para dispositivos que executam esse protocolo, esses dispositivos são chamados de refletores. Como o objetivo é que a resposta a essas requisições seja enviada para o alvo, o atacante deve fazer o *spoofing* do pacote IP, conforme descrito na próxima seção.

Um exemplo notável de um ataque por reflexão foi o abuso do protocolo Memcached no ataque ao GitHub em Fevereiro de 2018 [11]. Neste ataque, que usa servidores Memchached como refletores, foi alcançado um pico de 1,35 Tbps, o maior ataque de negação de serviço registrado até aquela época. Outros exemplos podem ser vistos em [9].

2.1.3 IP Spoofing

O IP *Spoofing* é o ato de utilizar como endereço de origem do pacote IP um endereço falso. Isto faz com que, apesar de o pacote ser entregue normalmente para o endereço destino, o destinatário envie respostas para o endereço falso. Além de ser necessário para realizar um ataque de reflexão, o *spoofing* de IP ajuda a esconder o atacante e dificultar o combate ao ataque, já que o uso de diversos refletores dificulta a filtragem do tráfego malicioso do legítimo.

A técnica de IP *spoofing* incorre na impossibilidade do atacante de receber respostas do refletor. Isto significa que é muito difícil IP *spoofing* em aplicações baseadas em TCP[1]. O TCP usa números de sequência, que são incrementados a cada envio, para garantir a entrega em ordem. Isto significa que, assumindo que o alvo já tenha iniciado a comunicação com o refletor, o atacante deverá descobrir o número de sequência sendo usado no momento do ataque. Um exemplo desta situação está ilustrada na Figura 2.1.

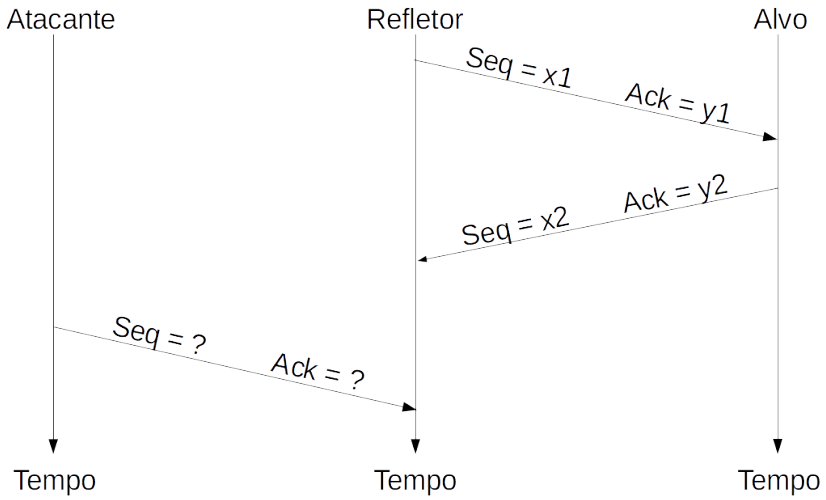

Figura 2.1: *Spoofing* em TCP com uma conexão previamente estabelecida.

Neste exemplo, o alvo e o refletor já fizeram transferência de dados entre si. Como tanto os valores de "x2" e "y2" quanto o volume de dados enviados são desconhecidos

[1]A especificação do TCP pode ser vista nas RFC 793 e 1122

pelo atacante, o envio de uma requisição com os números de sequência e ACK válidos é uma tarefa difícil.

O mesmo problema ocorre quando a conexão ainda não está estabelecida. Isto ocorre devido ao uso da técnica de *handshake* em 3 tempos para iniciar uma conexão TCP. A Figura 2.2 apresenta uma ilustração desses passos, são eles:

1. O cliente envia uma requisição para abrir uma conexão para o servidor. Para isso envia um segmento *Synchronize* (SYN) com um número de sequência aleatório.
2. O servidor então responde com um segmento SYN-ACK com número de sequência aleatório e o número de reconhecimento com valor igual ao número de sequência do segmento SYN do cliente acrescido em uma unidade.
3. O cliente então responde para o servidor com um *Acknowledgment* (ACK), com número de sequência igual ao número de reconhecimento do segmento SYN-ACK recebido e número de reconhecimento igual ao número de sequência do SYN-ACK recebido mais 1.

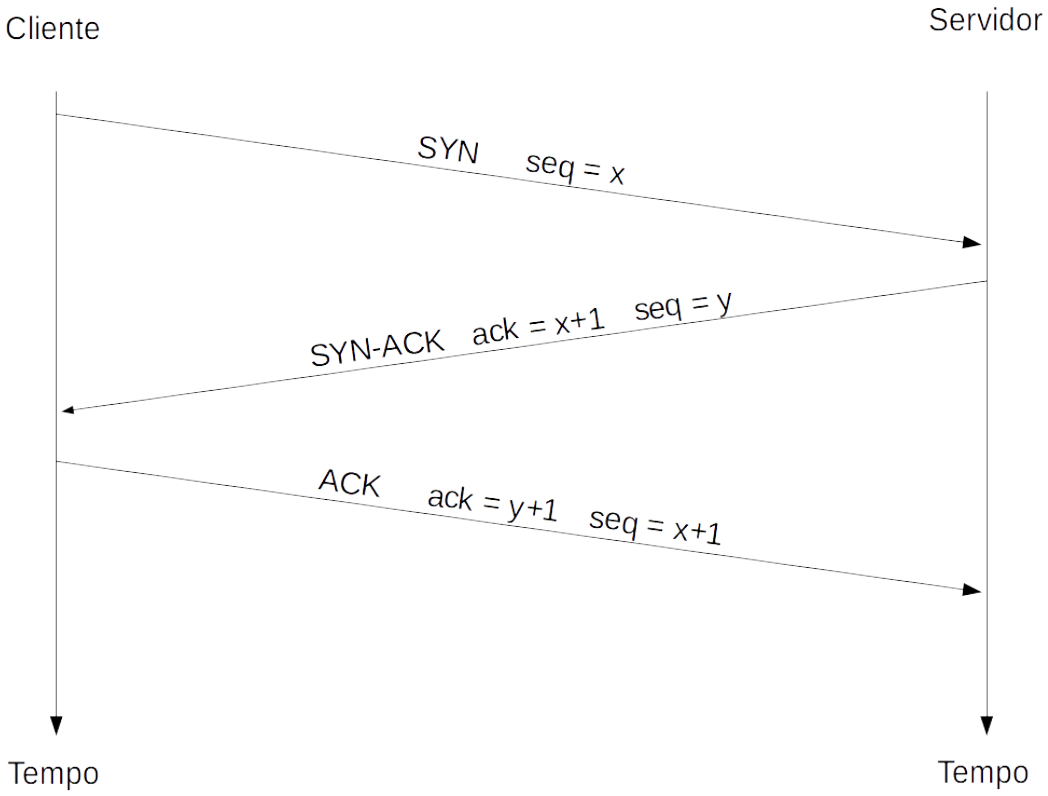

Figura 2.2: Estabelecimento de uma conexão TCP.

Note os valores dos números de sequência e de reconhecimento de cada mensagem, indicados respectivamente por “seq” e “ack”.

O problema está na determinação do valor do número de sequência do ACK do passo 3. Este número foi gerado com base no número de sequência gerado aleatoriamente pelo servidor no passo 2, e como o endereço falso é quem recebe as respostas do servidor, o atacante terá que adivinhar o número gerado para poder enviar o ACK do terceiro passo. Por estes motivos, os protocolos não orientados a conexão baseados em UDP são visados para os ataques de reflexão.

2.1.4 Ataques da Camada de Aplicação

Os ataques da camada de aplicação buscam esgotar os recursos das máquinas executando o serviço alvo, em contraste com ataques de camada 3 ou 4 que buscam congestionar a rede. Isto pode ocorrer tanto devido a uma inundação de requisições, um ataque volumétrico, quanto por um abuso de protocolo, onde o atacante explora uma falha que causa um grande consumo de recursos pelo servidor.

2.1.5 Ataques *Low and Slow*

Os ataques *Low and Slow* (*Low Volume and Slow Rate*) são ataques de baixo volume, em contraste com ataques volumétricos. Neste tipo de ataque, o atacante explora uma vulnerabilidade da implementação de um protocolo de forma a esgotar os recursos do servidor. Um exemplo é o ataque Slowloris [12] [13]. Este ataque realiza vários envios de requisições HTTP incompletas, de forma que o servidor mantenha várias conexões abertas, eventualmente esgotando seus recursos computacionais. Para isso, o ataque consiste em enviar partes da mensagem de requisição de forma lenta, para manter a conexão aberta o maior tempo possível.

2.1.6 Ataques Distribuídos de Negação de Serviço

Os ataques distribuídos, em inglês *Distributed Denial of Service* (DDoS), são caracterizados pelos múltiplos vetores de ataques. Com o ataque distribuído por vários agentes atacantes é possível alcançar uma vazão maior do que apenas uma máquina conseguiria. Um ataque volumétrico coordenado por grupos ativistas como Annonymous, em que várias pessoas são incitadas a congestionar um site, é um exemplo de um ataque distribuído de difícil mitigação já que as várias pessoas participantes são difíceis de distinguir do tráfego legítimo.

Um método comum de distribuir ataques é o uso de *botnets*. As *botnets* são redes de computadores que são coordenadas por um centro de controle. As *botnets* podem ser

formadas de forma voluntária, como por exemplo no uso do programa *Low Orbit Ion Cannon* (LOIC) na Operação *Payback*[2], ou de forma maliciosa, através da invasão de vários dispositivos devido a um *malware*, criando o chamado computador zumbi.

2.2 O Conjunto de Protocolos UPnP

O UPnP foi criado com objetivo de estender o conceito *Plug and Play* (PnP) para dispositivos em uma rede. Conceitualmente, um dispositivo com funcionalidade PnP pode ser conectado a outro sem manipulação de configurações, ou seja, um dispositivo PnP, após ser conectado fisicamente a outro, se adéqua automaticamente para ser utilizado sem necessidade de intervenção prévia do usuário. Em termos de redes de computadores, este tipo de tecnologia é conhecido como *zero-configuration networking* (zeroconf), onde um rede pronta para ser usada é criada sem intervenção do usuário ou pré configuração especializada. Além das facilidades providas por ser zeroconf, o UPnP também busca ser extensível e ao mesmo tempo independente de cada vendedor de hardware.

A especificação do UPnP é definida pelo documento conhecido como *UPnP Device Architecture* (UDA). Até a conclusão deste trabalho, a UDA possuía 3 versões:

1.0 : Versão inicial lançada em 2000. A última revisão foi em 2008;

1.1 : Pequena atualização da versão inicial, lançada em 2008;

2.0 : Versão atual, utilizada no UPnP+.

2.2.1 Distribuição de Versões

A versão UDA, o sistema operacional e a implementação sendo utilizadas por um dispositivo podem ser identificados através do campo "SERVER" de uma mensagem de descobrimento enviada em resposta a uma requisição de busca. Desta forma é possível identificar os dispositivos usando uma assinatura formada pela junção do endereço IP com os dados desse campo.

No relatório em [10] foram encontrados 93 milhões de assinaturas únicas, compostas pelo conjunto endereço IP e versão do servidor SSDP. Destes dispositivos, mais de 25% (23,6 milhões) usavam a *Software Development Kit* (SDK) da Intel [14] ou a biblioteca que continuou este projeto, conhecida como libupnp [15]. Outra biblioteca famosa, conhecida como MiniUPnP [16], é responsável por mais de 21% (19,4 milhões) das assinaturas. Ambas as implementações usam a especificação provida pela UDA versão 1.0 [17], mas no caso do MiniUPnP existe suporte parcial ao *Internet Gateway Device version 2* (IGDv2)

[2] `https://www.bbc.com/news/technology-11971259`

[18], que pode ser suportado tanto pela UDA 1.0 quanto a 1.1. Outras SDKs proprietárias que não possuem identificação própria, denominados Unknown SDK 1 e Unknown SDK 2 no relatório, provavelmente implementam a UDA 1.0, já que expõem a *string* UPnP/1.0 em seus cabeçalhos. Desta forma, conclui-se que a grande maioria dos dispositivos utiliza a UDA 1.0, uma versão cuja última revisão data de 15 de Outubro de 2008.

2.2.2 UPnP Device Architecture 1.0 (UDA 1.0)

Sendo a versão mais utilizada, este trabalho focará nesta versão específica, conforme a documentação oficial [17].

Em termos de comunicação, o UPnP define dois tipos de dispositivos. Dispositivos controlados, ou simplesmente "dispositivos", e pontos de controle. Como indica o nome, os dispositivos controlados ficam a espera de requisições dos pontos de controle. Note que um ponto final de comunicação pode conter mais de um dispositivo e ponto de controle, incluindo os dois tipos rodando simultaneamente.

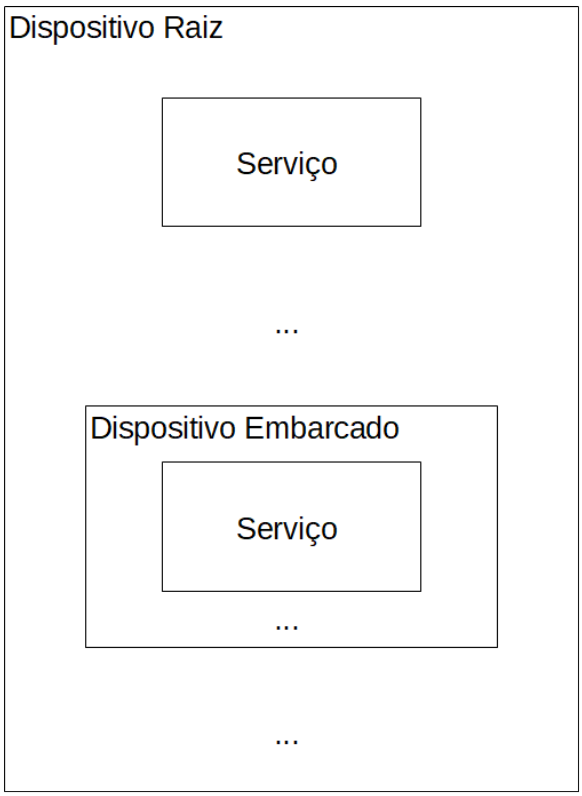

Figura 2.3: Estrutura lógica de um dispositivo UPnP.

Além disso, o UPnP define uma estrutura lógica dos dispositivos, conforme a Figura 2.3. Um dispositivo é dito raiz se não estiver embarcado em outro dispositivo. Caso contrário, o dispositivo é chamado de dispositivo embarcado. As funcionalidades de cada dispositivos são chamadas de serviços. Um único dispositivo físico pode conter uma combinação de diversos dispositivos raiz e embarcados, cada um com seus serviços.

O funcionamento do UPnP é separado em 6 passos. Após a obtenção de um endereço (Passo 0), os dispositivos podem ser descobertos por pontos de controles na rede (Passo 1). Encontrado um dispositivo, os pontos de controle podem então obter as informações relacionadas as capacidades do dispositivo (Passo 2). Com estas informações é possível realizar os passos de controle (Passo 3), eventos (Passo 4) e apresentação (Passo 5), que são passos independentes. Uma ilustração deste funcionamento pode ser vista na Figura 2.4.

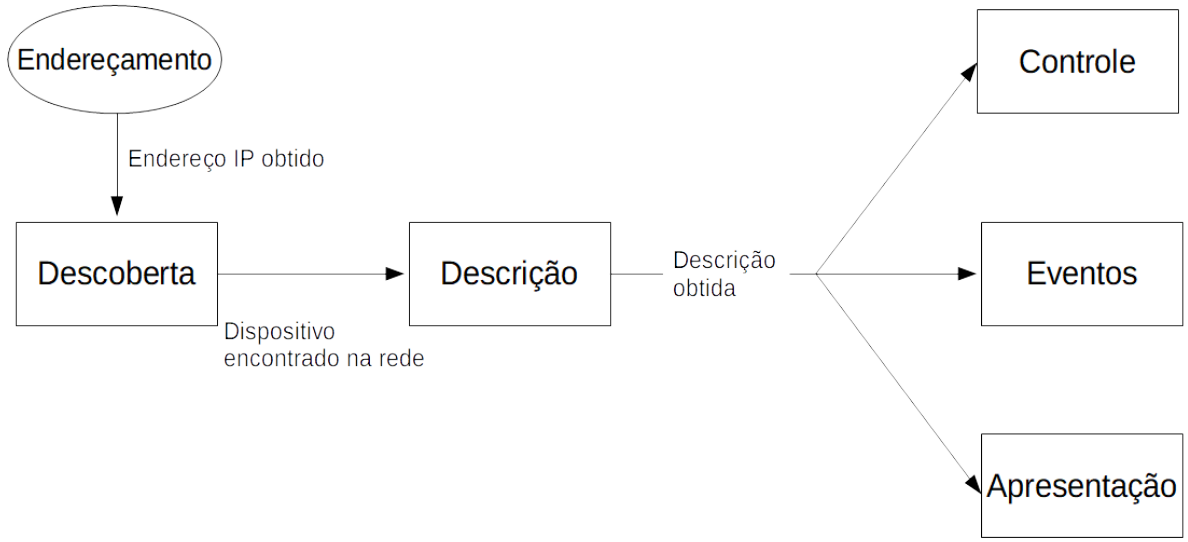

Figura 2.4: Passos do UPnP.

Passo 0: Endereçamento

O UPnP usa o protocolo IP como base, desta forma, o pré-requisito para o funcionamento de um dispositivo é a obtenção de um endereço IP. Este endereço pode ser obtido usando *Dynamic Host Configuration Protocol* (DHCP). Se o próprio dispositivo não implementa um servidor DHCP, então o dispositivo deve implementar um cliente DHCP. Caso não seja encontrado um servidor, o dispositivo deve então recorrer a uma autoconfiguração (Auto-IP), conforme recomendações da RFC 3927 [19].

Passo 1: Descoberta

Obtido um endereço IP, o próximo passo é o descobrimento. No UPnP este passo é realizado pelo protocolo SSDP, que é o principal alvo deste estudo. O protocolo SSDP usa uma variante *multicast* do HTTP que funciona sobre UDP. Conforme a especificação do HTTP[20], as mensagens são constituídas de um cabeçalho, onde cada campo de cabeçalho é separado por um pulo de linha (CRLF, *Carriage Return* e *Line Feed*) e formatado da seguinte forma:

```
Nome do campo: Valor
```

Note os espaços opcionais antes e depois de "Valor". A primeira linha excepcionalmente deve ser formatada da seguinte forma:

```
Tipo da mensagem * HTTP/1.1
```

Note que o asterisco é separado entre espaços. O texto do tipo da mensagem varia conforme os tipos explicados posteriormente. Deve-se incluir um pulo de linha (CRLF) adicional ao final do cabeçalho, o equivalente a adicionar uma linha em branco no final. Os campos de cabeçalho e o conteúdo do corpo da mensagem dependem do tipo de mensagem.

O passo de descoberta pode ser separado em duas categorias, anúncio e descoberta, conforme explicação a seguir.

Anúncio O anúncio ocorre durante 3 fases: quando o dispositivo é adicionado a rede, durante o período em que o dispositivo está disponível na rede e, no melhor caso, quando o dispositivo vai deixar a rede.

Quando um dispositivo é adicionado a rede, mensagens de descobrimento do tipo "NOTIFY" com campo "ssdp:alive" devem ser enviadas através do canal *multicast* `239.255.255.250:1900`, de forma a anunciar seu dispositivo raiz, seus dispositivos embarcados e seus serviços. O número de mensagens e o valor dos campos NT e USN, que identificam o tipo e a instância única deste tipo de dispositivo ou serviço, dependem da estrutura do dispositivo. A seguir é listado o número de mensagens enviadas. Mais detalhes no conteúdo de cada mensagem pode ser visto na especificação [17].

- 3 mensagens para o dispositivo raiz;
- 2 mensagens por dispositivo embarcado;
- 1 mensagem para cada tipo de serviço em cada dispositivo.

No total, para cada dispositivo raiz com d dispositivos embarcados e k tipos de serviços distintos, temos a seguinte fórmula:

$$N\acute{u}mero\_de\_Mensagens = 3 + 2d + k \tag{2.1}$$

A especificação recomenda enviar cada mensagem mais de uma e menos de três vezes devido a não confiabilidade do UDP. Além disso, estas mensagens de anúncio devem ser reenviadas antes do vencimento do período definido no campo CACHE-CONTROL, de modo a manter o registro de dispositivos atualizado em cada dispositivo. A especificação recomenda que tais envios sejam distribuídos aleatoriamente por um intervalo de menos da metade do período de vencimento.

O formato da mensagem é composto apenas do cabeçalho HTTP, conforme estrutura a seguir:

```
NOTIFY * HTTP/1.1
HOST: 239.255.255.250:1900
CACHE-CONTROL: max-age = segundos
LOCATION: URL
NT: Tipo de notificação
NTS: ssdp:alive
SERVER: SO/Versão UPnP/1.0 produto/versão
USN: UUID do anúncio
```

A descrição dos campos pode ser vista na Tabela 2.1.

Tabela 2.1: Descrição dos campos da mensagem NOTIFY.

Campo	**Descrição**
NOTIFY * HTTP/1.1	Determina o tipo da mensagem.
HOST	Indica o canal reservado pela IANA[3] para o SSDP.
CACHE-CONTROL	Determina o tempo em segundos da validade do anúncio.
LOCATION	Indica a URL para a descrição UPnP do dispositivo raiz.
NT	Identifica o tipo do dispositivo ou serviço que originou a mensagem. Detalhes dos valores aceitos por este campo podem ser vistos na documentação [17].
NTS	Especifica o subtipo da mensagem. Neste caso deve ser "ssdp:alive".
SERVER	Este campo identifica a implementação UPnP. Note que a parte "UPnP/1.0" identifica que o software implementa a UDA 1.0.
USN	Identifica unicamente cada instância de um dispositivo ou serviço. Detalhes dos valores aceitos por este campo podem ser vistos na documentação [17].

No caso de um dispositivo deixar a rede, uma mensagem do tipo NOTIFY com ssdp:byebye no campo NTS deve ser enviada para cada mensagem do subtipo ssdp:alive que tenha sido enviada e não esteja expirada. Caso o envio dessa mensagem não seja possível, o próprio vencimento das mensagens garante a sua remoção nos pontos de controle. A mensagem segue o seguinte formato, que consiste do cabeçalho HTTP também sem corpo:

```
NOTIFY * HTTP/1.1
HOST: 239.255.255.250:1900
NT: Tipo de notificação
NTS: ssdp:byebye
USN: UUID do anúncio
```

Os valores de NT e USN devem coincidir com os valores da mensagem ssdp:alive a que a mensagem "ssdp:byebye" se refere.

Busca O UPnP permite que pontos de controle enviem mensagens de busca para descobrir outros dispositivos na rede de forma imediata, sem ter que esperar pelas mensagens de anúncio. Para isso, deve-se fazer *multicast* de uma mensagem M-SEARCH no canal reservado para o SSDP(endereço IP `239.255.255.250` e porta 1900). Assim como a mensagem de anúncio, a mensagem de busca possui apenas o cabeçalho seguido de uma nova linha (CRLF) adicional, conforme o seguinte formato:

```
M-SEARCH * HTTP/1.1
HOST: 239.255.255.250:1900
MAN: "ssdp:discover"
MX: segundos
ST: alvo
```

A descrição dos campos pode ser vista na Tabela 2.2.

Tabela 2.2: Descrição dos campos da mensagem M-SEARCH.

Campo	**Descrição**
M-SEARCH * HTTP/1.1	Indica que estamos utilizando o método M-SEARCH, ou seja, se trata de uma mensagem de busca.
HOST	Assim como no método NOTIFY, o campo HOST denota o canal reservado para o SSDP.
MAN	Campo requerido pelo HTTP *Extension Framework*[4] e deve ser "ssdp:discover" (note que as aspas duplas são necessárias).
MX	Define o tempo máximo em segundos para os dispositivos responderem a requisição. Os dispositivos devem responder em um tempo aleatório entre 0 e o tempo definido por este campo, de forma a balancear a carga no ponto de controle. Recomenda-se um tempo entre 1 e 120 segundos inclusos.
ST	Define o escopo da busca. Pode ser "ssdp:all" para indicar todos os dispositivos e serviços ou valores que casem com os valores do campo NT no método NOTIFY.

Um dispositivo que receba uma mensagem M-SEARCH e esteja no escopo definido por ST deve responder seguindo o mesmo padrão da mensagem NOTIFY com ssdp:alive (ou seja, com várias respostas dependendo da estrutura do dispositivo), mas com campo ST no lugar do NT, conforme o seguinte formato :

```
HTTP/1.1 200 OK
CACHE-CONTROL: max-age = segundos
DATE: quando a resposta foi gerada
EXT:
LOCATION: URL
SERVER: OS/versão UPnP/1.0 produto/ versão
ST: escopo
USN: UUID
```

A resposta também não possui corpo. A definição dos campos pode ser vista na Tabela 2.3.

Tabela 2.3: Descrição dos campos da mensagem de resposta ao M-SEARCH.

Campo	**Descrição**
HTTP/1.1 200 OK	Indica sucesso da operação de busca, conforme especificado pelo protocolo HTTP.
CACHE-CONTROL	Especifica a validade do anúncio.
DATE	Campo recomendado que indica quando a resposta foi gerada.
EXT	Requerido pelo HTTP *Extension Framework*.
LOCATION	Contém a URL para a descrição do dispositivo raiz.
SERVER	Indica o sistema e a implementação sendo executada.
ST	Escopo da busca, conforme a mensagem de busca e seguindo a mesma lógica do campo NT da mensagem de anúncio.
USN	Identifica unicamente cada instância de um dispositivo ou serviço. Detalhes sobre a identificação podem ser vistos na documentação[17].

Passo 2: Descrição

Após a descoberta de um dispositivo, o próximo passo é obter mais informações dele. Para isso um ponto de controle envia uma requisição HTTP GET para a URL indicada pelo campo LOCATION das mensagens de descobrimento. O dispositivo então retorna um arquivo *Extensible Markup Language* (XML) [5] contendo a descrição do dispositivo, o que inclui informações sobre os dispositivos embarcados e URLs para as descrições dos serviços. As descrições dos serviços são obtidas da mesma maneira e também usam sintaxe XML. Mais informações e a formatação das descrições estão disponíveis em [17].

Passo 3: Controle

De posse da descrição do dispositivo, um ponto de controle pode então invocar ações que o serviço provê ou então consultar o valor das variáveis que modelam o estado em tempo de execução do serviço. Para invocar ações o ponto de controle envia mensagens de controle e em seguida espera uma mensagem contendo o resultado da ação. No caso da consulta de variáveis o ponto de controle envia uma mensagem de requisição e o dispositivo responde com o valor da variável. Este passo de controle é implementado seguindo o protocolo *Simple Object Access Protocol* (SOAP) [21], que por sua vez é uma extensão do HTTP, seguindo o HTTP *Extension Framework* [22].

Passo 4: Eventos

O UPnP permite que os pontos de controles possam ser avisados de alterações nas variáveis descritas no passo anterior. Para isso é usado um protocolo próprio para eventos, cha-

[5]Especificação: `https://www.w3.org/TR/2000/REC-xml-20001006`

mado *General Event Notification Architecture* (GENA)[6]. Assim como o SOAP, o GENA funciona sobre o protocolo HTTP. No contexto de eventos, um assinante é um ponto de controle interessado em receber eventos, e um dispositivo que possua variáveis que são monitoradas é dito editor. A informação de quais variáveis são monitoradas está presente na descrição do dispositivo obtida no passo anterior.

Um ponto de controle que deseje ser um assinante deve enviar uma mensagem de assinatura, opcionalmente com um requerimento de duração da assinatura. Se aceitar a assinatura, o editor deve responder com uma mensagem com a duração da assinatura. Finalmente, o editor envia um evento inicial contendo os valores de todas as variáveis monitoradas, de forma que o assinante possa inicializar sua modelagem do estado do serviço. Note que o assinante recebe todas as mensagens de eventos sobre todas as variáveis que são monitoradas, não há como se inscrever para receber eventos de variáveis específicas. Para manter a assinatura o ponto de controle deve mandar mensagens para renová-la antes que a duração da assinatura atual termine.

Para finalizar uma assinatura o ponto de controle deve enviar uma mensagem de cancelamento. Se por algum motivo essa mensagem não puder ser enviada, a assinatura será expirada naturalmente com o tempo, garantindo a remoção do assinante da lista do editor.

Passo 5: Apresentação

O passo de apresentação é a interface com o usuário. Nela é possível controlar ou ver o estado do dispositivo. Este passo é simplesmente implementado usando o protocolo HTTP. O ponto de controle envia um HTTP GET para a URL indicada na descrição do dispositivo, contida na mensagem de descrição do passo 2, recebendo então uma página no formato HTML. A única restrição imposta pelo UPnP é que a página utilize uma versão HTML maior ou igual a 3.0. Desta forma, a página apresentada no navegador do ponto de controle atua como uma interface de usuário.

2.3 O Ataque com Protocolo SSDP

Esta seção descreve as características que propiciam o ataque, além de descrever o ataque em si.

[6]O protocolo GENA é especificado na seção 4 da especificação UDA 1.0 [17]

2.3.1 Potencial de Amplificação

Conforme visto na explicação do protocolo SSDP, um ponto de controle pode a qualquer momento enviar uma requisição M-SEARCH para buscar dispositivos na rede. Cada dispositivo então envia mensagens correspondentes ao dispositivo raiz, seus dispositivos embarcados e seus serviços. A resposta além de gerar mensagens maiores que uma M-SEARCH, gera mais de uma mensagem, já que se espera que um dispositivo tenha minimamente um serviço, o que gera uma mensagem para o dispositivo raiz e outra para o serviço, evidenciando o potencial para amplificação.

2.3.2 Balanceamento de Carga

Protocolos inteligentes podem implementar algum tipo de controle de forma a evitar o congestionamento da rede. O SSDP utiliza o campo MX da mensagem M-SEARCH para indicar o tempo máximo para envio da resposta, conforme explicado no passo de descobrimento. Apesar de haver uma recomendação para que o MX seja no mínimo 1, a própria especificação dá liberdade para que os dispositivos possam assumir valores menores que os especificados pelo MX. Isto significa que é plausível a geração de um tráfego significativo. Outro ponto a se notar é que apesar do campo MX ser obrigatório, o comportamento do dispositivo respondente em relação ao valor deste campo não é mandatório. Desta forma, a capacidade de gerar uma amplificação sustentável depende da implementação. Conforme relatado em [23], grande parte dos refletores utiliza as mesmas implementações indicadas em [10]. Conclui-se então que estas implementações seguem a recomendação quanto a interpretação do valor MX.

2.3.3 Reflexão

A possibilidade de IP *spoofing* de forma que o refletor envie suas respostas para o alvo é o ponto chave para a reflexão. O protocolo SSDP é baseado no UDP e não é orientado a conexão, o que significa que o IP *spoofing* é efetivo. O problema entretanto está no fato de que a documentação descreve que as mensagens de busca são enviadas via *multicast*. Este tipo de mensagem, quando enviado por um usuário comum para uso genérico, é ignorada pelos roteadores que formam o *backbone* da internet. Desta forma, se a implementação UPnP verifica o IP destino da mensagem M-SEARCH para checar se o endereço é o do grupo *multicast*, uma mensagem recebida via *unicast* deveria ser ignorada pelo dispositivo. Entretanto, como tal checagem gera um processamento adicional, as implementações geralmente não fazem essa checagem. Isto é demonstrado conforme [23] e [10], onde foi possível receber um grande número de respostas.

É importante notar que as especificações mais recentes, UDA versões 1.1 e 2.0, preveem o uso de um tipo de mensagem de busca que pode ser enviado via *unicast*. Como neste tipo de mensagem não há o campo MX, a especificação recomenda que o dispositivo envie uma resposta em até 1 segundo. Desta forma o ataque é facilitado.

2.3.4 Metodologia do Ataque

O ataque usando a mensagem M-SEARCH se dá conforme os seguintes passos:

1. É gerada uma mensagem M-SEARCH com campo MX com valor igual a 1, o menor valor recomendado, e ST com valor igual a "ssdp:all", para que a mensagem se aplique a todos os dispositivos UPnP na rede;
2. É criado um datagrama UDP com a mensagem e com porta de destino 1900, que é a porta reservada para o SSDP;
3. É criado um pacote IP com endereço de origem falsificado, para coincidir com o endereço do alvo, e endereço de destino igual ao do refletor. O datagrama UDP então é colocado no pacote IP;
4. O pacote IP é enviado para o refletor, que pode ser qualquer dispositivo UPnP que responda a mensagens de descobrimento vindas de fora da rede local;
5. O refletor recebe a mensagem e envia inadvertidamente a resposta para o alvo.

2.4 O Ataque com Protocolo SNMP

Esta seção descreve as características do protocolo SNMP relacionadas ao ataque, que é realizado através do método *GetBulkRequest*.

O SNMP é um protocolo de gerência de rede, onde existem os dispositivos gerenciados e as estações de gerenciamento. Os nós gerenciados mantêm uma base de dados, chamada *Management Information Base* (MIB), que contêm variáveis que representam os dados de gerência. Estas variáveis podem ser usadas tanto para monitoramento quanto para invocar ações em nós gerenciados. As requisições para consulta ou mudança de variáveis é feita através de mensagens enviadas via UDP ou TCP na porta reservada 161.

2.4.1 Reflexão

Nas versões 1 e 2c, a autenticação das mensagens é baseada no valor do campo *"community"*, que é codificada como texto pleno e transmitido às claras. Não o bastante, muitas implementações não alteram a comunidade *"public"* que vem como padrão de fábrica.

Como o SNMP suporta mensagens enviadas via UDP, o spoofing do pacote IP é trivial. Desta forma, usando uma mensagem SNMP com IP *spoofing* e com campo *"community"* igual a *"public"* é possível que um nó gerenciado envie uma resposta para o alvo do ataque.

2.4.2 Amplificação

Na versão 2 e derivadas foi implementado uma nova mensagem de requisição *GetBulkRequest*, que serve para pegar o valor de várias linhas de uma variável tabular em apenas uma mensagem. Uma variável tabular é aquela que pode possuir várias iterações, que representam uma linha de uma tabela. Caso o valor do argumento *"max-repetitions"* da mensagem *GetBulkRequest* ultrapasse o número de linhas da tabela, as próximas variáveis da MIB são retornadas. Desta maneira, um atacante pode enviar um *GetBulkRequest* com um número alto de *"max-repetitions"*, gerando uma mensagem de resposta com tamanho significativo.

2.4.3 Balanceamento de Carga

Ao contrário do SSDP, o SNMP não possui nenhuma recomendação quanto ao intervalo de respostas. Entretanto, devido ao grande custo de processamento do *GetBulkRequest*, o refletor pode não conseguir enviar as respostas imediatamente.

2.4.4 Metodologia do Ataque

O ataque se da conforme os seguintes passos:

1. É gerado uma requisição *GetBulkRequest* com o identificador de uma variável tabular. O campo *"non-repeaters"* deve ser 0, o que indica que não há identificadores de variáveis escalares na mensagem. O valor do campo *"max-repetitions"* deve ser um valor alto mas que não faça a resposta ultrapassar o tamanho limite definido pelo *Internet Protocol version 4* (IPv4), ou seja, 65.535 bytes;
2. É gerado o datagrama UDP da mensagem, com porta igual 161, que é reservada para o SNMP;
3. É gerado o pacote IP com endereço de origem igual ao do alvo e endereço de destino igual ao do refletor;
4. O datagrama UDP é colocado no pacote IP e enviado normalmente para o refletor;
5. O refletor recebe a requisição e prepara uma resposta com uma mensagem contendo as variáveis requisitadas;

6. Finalmente, o refletor envia a mensagem para o alvo.

2.5 Mitigação dos Ataques

Existem três frentes que podem ser abordadas para mitigar os ataques.

2.5.1 *Spoofing* de IP

Um dos fatores que possibilita a reflexão é o *spoofing* de IP. As práticas descritas na BCP 38[7] recomendam que roteadores façam a filtragem do tráfego de entrada, de forma a bloquear pacotes com IP de origem que não correspondam a sub-rede ao qual pertencem, o que eliminaria a possibilidade de fazer *spoofing* do IP.

2.5.2 Vulnerabilidades dos Protocolos

Muitas vezes, aspectos da arquitetura ou da implementação do protocolo podem facilitar ou permitir os ataques.

No caso do SSDP o problema está na mal implementação de alguns roteadores domésticos. O UPnP foi desenvolvido para uso em redes privadas, ou seja, os roteadores não deveriam aceitar requisições ou enviar respostas na porta WAN. Seria impossível realizar o ataque caso essa regra fosse seguida.

No caso do SNMP existem diversos fatores que contribuem para o ataque:

- O uso da *community string* padrão *"public"* permite o atacante a enviar requisições válidas de leitura para o dispositivo gerenciado. A simples mudança ou remoção desta comunidade dificultaria o ataque, já que o atacante terá que descobrir uma comunidade com permissão de leitura, de modo que suas requisições sejam aceitas.
- O uso do SNMPv1 e v2c significa que o valor da *community string* é transmitido às claras, o que facilita a sua descoberta, possibilitando o envio de requisições válidas pelo atacante. A versão mais nova, SNMPv3, permite o uso de criptografia, o que dificultaria o ataque.
- Os nós gerenciados podem se limitar a aceitar apenas mensagens de IP de origem correspondente às estações de gerenciamento ou redes conhecidas. Apesar de não evitar outros problemas de segurança devido ao *spoofing* de IP, essa medida limitaria o ataque de negação de serviço.

[7] `https://tools.ietf.org/html/bcp38`

2.6 Considerações Finais

Com os achados deste capítulo, conclui-se que ambos os ataques com SSDP e SNMP são viáveis e de relativa facilidade de se executar.

As escolhas dos campos ST e MX do protocolo SSDP e do campo *"max-repetitons"* do SNMP são fundamentais para a eficiência do ataque.

Com as informações deste capítulo, buscou-se implementar uma ferramenta capaz de detectar refletores e realizar ambos os ataques de forma eficiente. Esta ferramenta está descrita no Capítulo 3.

Capítulo 3

A Ferramenta

Este Capítulo descreve a arquitetura da ferramenta, os diferenciais em relação a ferramenta original e a implementação do ataque com protocolo SSDP. Detalhes da implementação do protocolo SNMP estão em [1].

3.1 Arquitetura

A arquitetura da ferramenta permanece a mesma da ferramenta original, com os módulos GUI, Scanner e Striker. Conforme ilustra a figura usada por Medeiros Figura 3.1, o módulo GUI utiliza os serviços dos módulos Scanner e Striker:

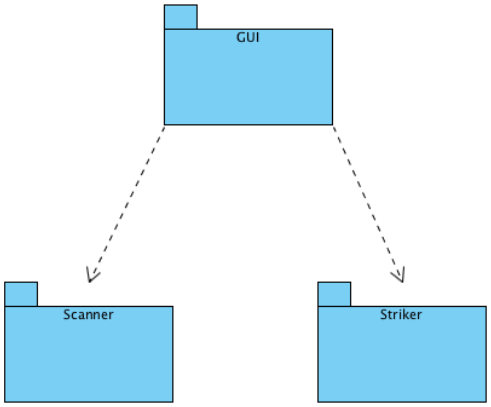

Figura 3.1: Módulos da ferramenta (Fonte: [1]).

O programa foi feito usando a linguagem Java e C. A linguagem Java foi escolhida devido à portabilidade e a possibilidade de usar a biblioteca padrão Swing para construção de interfaces gráficas, o que dispensa o uso de bibliotecas externas extras. A linguagem

C foi usada devido à necessidade de se usar os chamados *raw sockets*, o que é necessário para se fazer o spoofing do pacote IP.

3.1.1 GUI – Módulo de Interface do Usuário

O módulo GUI é responsável por facilitar a passagem dos parâmetros do usuário para os serviços dos módulos Scanner e Striker. A GUI é composta por duas janelas: uma janela principal e uma de escaneamento.

A Janela Principal

A janela principal é responsável por mostrar os resultados obtidos pelo Scanner e por configurar e realizar o ataque.

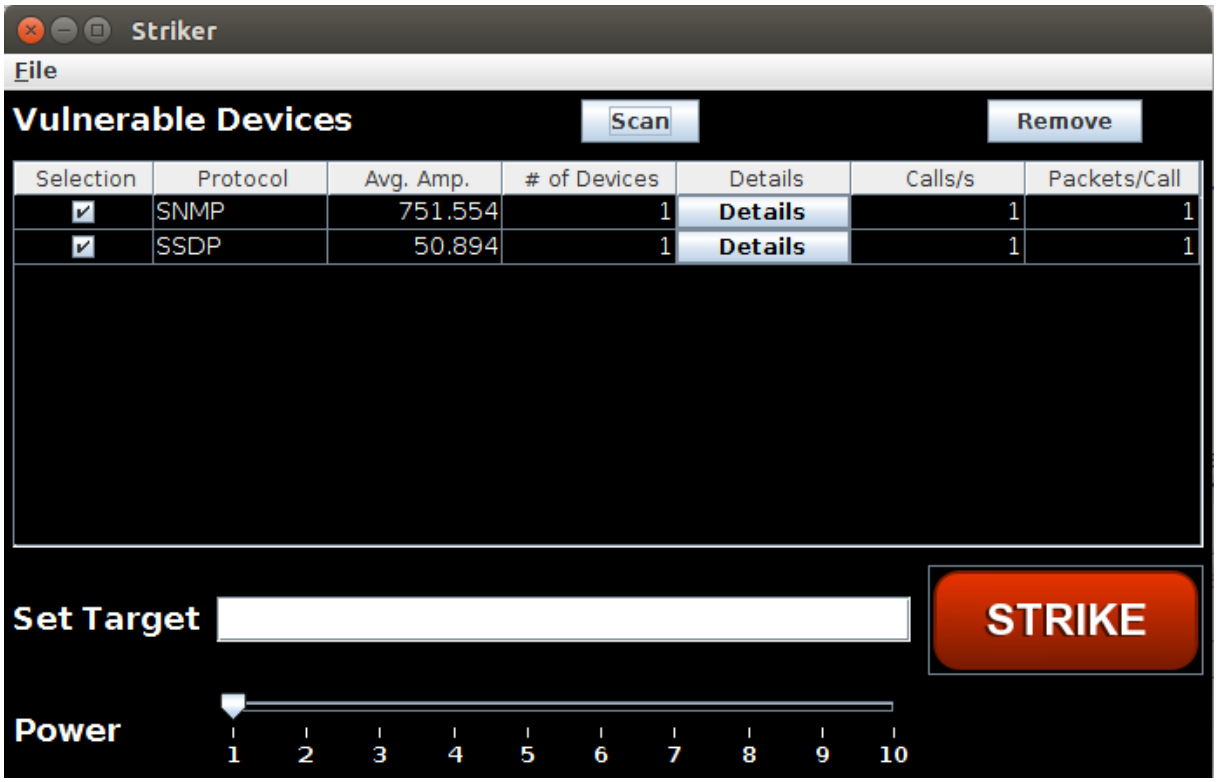

Figura 3.2: A janela principal da ferramenta.

Esta janela possui as seguintes funcionalidades:

Importar/Exportar: O menu "*File*" na parte superior da janela possibilita importar e exportar refletores encontrados na janela de escaneamento. A ferramenta faz o uso do Gson, uma biblioteca Java do Google [24], para converter os dados para o formato *JavaScript Object Notation* (JSON)[1];

[1]RFC 8259: `https://tools.ietf.org/html/rfc8259`

Tabela de Refletores: A tabela no centro da janela mostra a lista de refletores de cada protocolo. O botão "*Details*" mostra a lista dos dispositivos refletores e o botão "*Remove*" remove as listas marcadas. A Figura 3.3 mostra a aparência da janela que aparece ao cliar no botão "*Details*";

Configuração do Ataque O campo "*Selection*" da tabela possibilita selecionar listas de dispositivos de protocolos específicos para o ataque. A taxa de envio pode ser controlada usando o controlador deslizante "*Power*", para uma configuração fácil, ou modificando os campos "*Calls/s*" e "*Packets/Call*" da tabela, para uma configuração manual. O campo "*Set Target*" determina o endereço IP do alvo. O botão "*Strike*" dá início ao ataque;

Escaneamento: O botão "*Scan*" abre a janela de escaneamento.

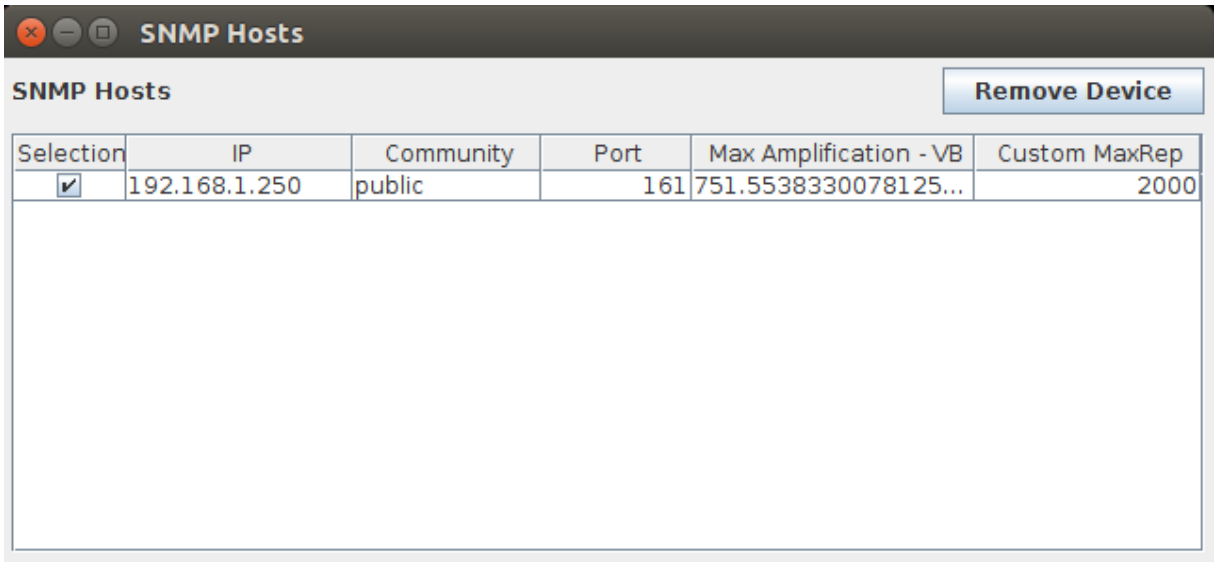

Figura 3.3: Janela com a lista de refletores.

A Janela de Escaneamento

A janela de escaneamento faz a interface com o módulo Scanner. Nela é possível configurar os parâmetros e realizar o escaneamento.

O campo "Base IP" associado com os botões de rádio são usados para determinar os alvos do escaneamento. Com eles é possível escolher um único IP ou uma sub-rede inteira.

A caixa de combinação é usada para selecionar o protocolo. Dependendo da seleção, outras opções específicas de cada protocolo aparecem abaixo.

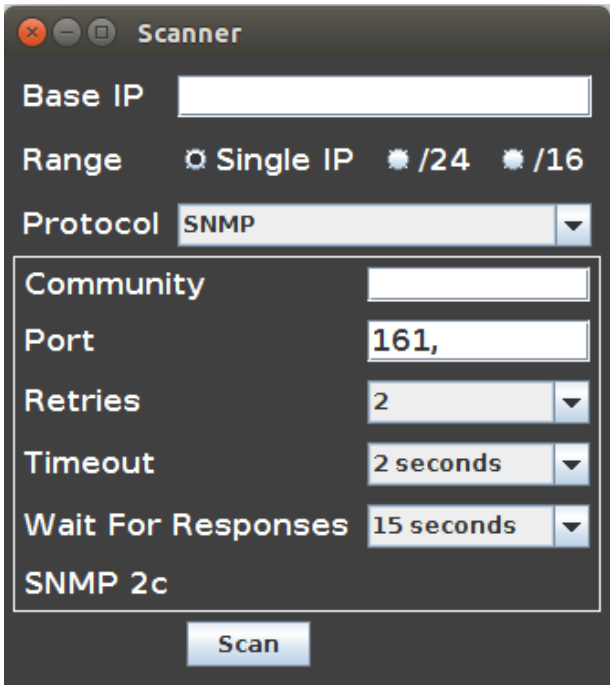

Figura 3.4: Janela de escaneamento com SNMP selecionado.

No caso do SNMP temos as seguintes opções:

Tabela 3.1: Opções de escaneamento para o SNMP.

Opção	**Descrição**
Community	Define o valor do campo *"community"* da mensagem SNMP.
Port	Define a porta de destino.
Retries	Define o número de tentativas extras de envios de requisições.
Timeout	Define o tempo limite para espera de respostas.
Wait For Responses	Define o tempo máximo total de espera. Metade do tempo é usado no descobrimento de dispositivos e a outra metade no cálculo da amplificação.

Para o SSDP temos as seguintes opções:

Tabela 3.2: Opções de escaneamento para o SSDP.

Opção	**Descrição**
Retries	Define o número de tentativas extras de envios de requisições.
Timeout	Define o tempo de espera para recebimento de respostas antes de iniciar uma nova tentativa.

Ao final da janela temos o botão "*Scan*" que inicia o processo de escaneamento.

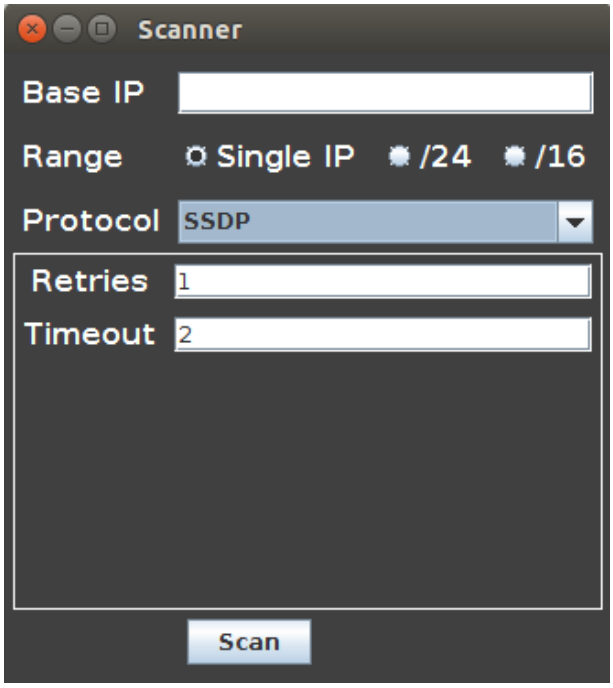

Figura 3.5: Janela de escaneamento com SSDP selecionado.

3.1.2 Scanner – Módulo de Escaneamento

O Scanner foi modelado de forma que cada protocolo funcione como um módulo que segue as regras de uma interface chamada "Finder". Seguindo a interface, cada módulo deve ter um método para iniciar a busca de refletores e outro método para pegar os refletores obtidos. Os refletores foram modelados usando uma classe base chamada IVulnerableHost, que deve ser estendida para cada protocolo. Graças a este modelo, o módulo Scanner é desacoplado da GUI, o que facilita a implementação de novos protocolos.

3.1.3 Striker – Módulo de Ataque

O Striker foi modelado usando a classe Java Striker e a biblioteca nativa em C. A ligação entre as duas partes se dá através do *Framework* JNI [25], que atua como interface entre o código Java e o código nativo.

A classe Striker

Esta classe implementa as *threads* de ataque, que fazem chamadas a função nativa C que envia as requisições. A Striker regula o número de chamadas a função C através da classe *Rate Limiter*, presente no projeto Guava [26] do Google.

A biblioteca Nativa C

A biblioteca nativa cria os pacotes IP e os datagramas UDP de acordo com as informações passadas pela classe Striker. O envio das mensagens é feita usando *raw sockets*, o que é necessário para se possa construir os pacotes IP manualmente. Isso é necessário para que se possa colocar o endereço do alvo no campo de destino do pacote IP. O uso de *raw sockets* faz com que a biblioteca em C seja o único componente que não é portável. Neste trabalho, a biblioteca C foi compilada para sistemas Linux, mas é possível que bibliotecas para outros sistemas sejam implementadas, bastando apenas seguir a assinatura usada pela classe Striker.

3.2 Alterações da Ferramenta Original

Nesta seção são apresentados os detalhes da implementação que foram modificados em relação a ferramenta original. A maior parte das modificações foram feitas para possibilitar e facilitar a implementação de novos protocolos na ferramenta.

3.2.1 Alterações na Implementação da GUI

Uso de Gerenciadores de Leiaute

A GUI original usava posicionamento absoluto dos componentes. Isto significa que o programador deve calcular o posicionamento de novos elementos o que pode ser muito tedioso. Além disso, o posicionamento e o tamanho dos componentes era fixo, de forma que não era possível ajustar o tamanho das janelas. Consequentemente, se o usuário usar uma tela de baixa resolução, é possível que alguns elementos da GUI sejam truncados. A nova GUI utiliza o gerenciador padrão "GridBagLayout", que remove todas as limitações da GUI antiga.

Ainda visando a facilidade de implementar novos protocolos, a parte específica de cada protocolo da janela de escaneamento faz uso do gerenciador "CardLayout". Este gerenciador permite que vários agrupamentos de componentes gráficos compartilhem o mesmo espaço. Para isso, cada grupo é modelado como uma carta que divide um espaço onde apenas uma carta pode estar visível. Para implementar novos protocolos basta então adicionar uma carta com os campos específicos do protocolo.

Alterações das Configurações

Na janela de escaneamento original, o usuário poderia escolher um único IP ou uma sub-rede completa. Entretanto, a escolha de um IP único implicava em não selecionar

nenhum dos botões de rádio do campo "*Range*", que definem o comprimento do prefixo de roteamento. Isto significa que, após selecionar uma das opções do campo "*Range*", não é possível voltar atrás e usar a opção de IP único. Isto foi corrigido com a adição de outro botão de rádio ao conjunto, que permite o uso de um único endereço IP.

Na janela principal original era possível escolher a potência do ataque usando um controlador deslizante, que dividia a potência em dez níveis, conforme o seguinte cálculo:

$$RPS = 10^{pot-1} \tag{3.1}$$

Onde "RPS" é o número de requisições por segundo e "pot" é o nível escolhido. O modo como as requisições são enviadas dependem da potência escolhida. No nível 1, a chamada a função de envio é limitada a 1 vez por segundo e cada chamada faz o envio de 1 pacote. A partir do nível 2, as chamadas a funções são sempre limitadas a 10 chamadas por segundo e o número de pacotes enviados por chamada vai aumentando a cada nível. Apesar desta abordagem prover um método simples para controle de potência, ela não permite uma customização granular da taxa de envio. Por isso, na nova implementação foram incluídos os campos "*Calls/s*" e "*Packets/Call*", que permitem a configuração manual destes parâmetros.

Alterações das Tabelas

Originalmente a janela principal exibia uma tabela com os refletores SNMP. Como a nova implementação permite a inclusão de novos protocolos, este esquema teve que ser alterado. A janela principal agora mostra uma tabela com as listas de refletores de cada protocolo. Para manter a granularidade anterior, é possível clicar no botão "*Details*" para que seja mostrada uma tabela que lista cada refletor da lista associada ao botão, conforme mostra a Figura 3.3.

3.2.2 Alterações na Implementação do Scanner

As principais modificações no Scanner são o uso da interface Finder e da criação da classe abstrata IVulnerableHost.

A Interface Finder

Na implementação original, a classe Finder servia para desacoplar a GUI do módulo Scanner. Entretanto, a implementação era restrita ao SNMP. A nova implementação transformou a Finder em uma interface onde classes que desejem implementá-la devem fazer a implementação dos métodos abstratos, que funcionam como uma comunicação

entre os módulos GUI e Scanner. Desta forma, a implementação específica de cada protocolo na GUI e na Scanner ficam desacopladas não apenas para o protocolo SNMP, mas também para futuras implementações.

A Classe IVulnerableHost

Ainda visando a padronização entre os diferentes protocolos, foi criado a classe abstrata IVulnerableHost, que funciona como a base das classes que modelam os dispositivos refletores de cada protocolo. Esta classe possui os métodos e atributos comuns a todos os protocolos. A classe original usada para o SNMP foi alterada para estender esta classe. Desta forma, promove-se o reúso de código e reduz-se o acoplamento entre a GUI e a Scanner.

3.2.3 Alterações na Implementação da Importação e Exportação

Na implementação original, a exportação e importação dos dados fazia a conversão entre objetos Java e o arquivo JSON usando as funções padrões da biblioteca Gson [24]. Na nova implementação, a lista de refletores ganhou uma nova hierarquia, representando os diferentes protocolos. Isto significa que a lista de refletores se transformou em uma lista de protocolos, que por sua vez possuem uma lista de refletores. Como o uso de polimorfismo não é suportado na versão atual do Gson, foi necessário implementar adaptadores customizados para a conversão entre JSON e os objetos Java.

3.3 Implementação do Módulo SSDP

O módulo SSDP é responsável por fazer o escaneamento e retornar a lista de refletores SSDP encontrados. Este módulo é implementado por duas classes, Finder_SSDP e Listener_SSDP.

A classe Finder_SSDP implementa a interface Finder, ou seja, pega os dados passados pela GUI para executar um escaneamento, retornando uma lista de refletores. Para fazer o envio das requisições e escutar as respostas esta classe utiliza um sistema de *pool* de *worker threads* de tamanho fixo. A classe Finder_SSDP funciona como o supervisor das *threads* e a classe Listener_SSDP funciona como a *worker thread*. Cada endereço IP passado para a Finder_SSDP resulta na criação de uma *thread* Listener_SSDP, que tentará fazer requisições para o possível refletor.

A classe Listener_SSDP faz o envio das mensagens M-SEARCH. Para aumentar a abrangência da ferramenta, a classe inicialmente tenta fazer o envio da mensagem na

formatação especificada pelas versões UDA 1.1 e posteriores. Se não houver resposta, a classe tenta então o envio na formatação UDA 1.0. O número de tentativas para cada versão e o intervalo entre as tentativas são determinados pelos parâmetros passados pelo usuário. Se for obtido uma resposta, a classe então cria um objeto VH_SSDP, que é a extensão SSDP da classe IVulnerableHost, com os valores do endereço IP e da amplificação obtida. Finalmente, esse objeto é retornado para a Finder_SSDP.

3.4 Comparativo com Outras Ferramentas

A maior parte das ferramentas disponíveis publicamente consiste de pequenos *scripts* que simplesmente realizam ataques, sobre um único protocolo, usando uma lista de refletores provida pelo usuário. Em relação a essas ferramentas mais simples, a desenvolvida neste trabalho possui o diferencial de acoplar tanto o programa escaneador quanto o que executa o ataque em si. Além disso, o uso de uma interface gráfica pode facilitar o uso da ferramenta para uma maior gama de usuários. Outro diferencial é a possibilidade de realizar ataques com mais de um protocolos, de forma simultânea e dentro de uma única interface.

Ferramentas mais complexas e com melhor usabilidade geralmente são de uso privado e ficam sob sigilo seja por interesses comerciais, como no caso de empresas de segurança, ou por questões legais, como é o caso de muitos artigos acadêmicos.

Uma ferramenta pública muito conhecida é a Saddam [27], que pode realizar ataques com DNS, NTP, SNMP e SSDP. Apesar de possuir uma boa gama de ataques, esta ferramenta não possui a funcionalidade de escanear refletores nem a possibilidade de configurar a potência do ataque.

3.5 Considerações Finais

Este capítulo descreveu a arquitetura da ferramenta, os casos de uso, as modificações realizadas e a implementação do ataque com SSDP. A maior parte das modificações foi feita com a intenção de facilitar a implementação de novos protocolos na ferramenta, de forma que trabalhos futuros com outros protocolos possam utilizá-la como base.

Capítulo 4

Testes e Resultados

Neste capítulo são apresentados os resultados obtidos com os testes de simulação de ataques. O ambiente de testes consiste de 4 dispositivos:

Atacante: O atacante é o dispositivo que executa a ferramenta. Para o atacante foi usado o notebook Dell Inspiron 14R 5437, com sistema operacional Windows 10 rodando uma máquina virtual Ubuntu 14.04 LTS. A configuração é a seguinte:

- Processador i7-4500U @ 1,80GHz
- 8GB de memória RAM DDR3L @1600MHz
- Adaptador de rede RTL8136
- Sistema hospedeiro Windows 10 Home x64 Versão 1803
- Programa de virtualização VirtualBox 5.2.18
- Sistema convidado Ubuntu 14.04 (64-bit)

Refletor: O refletor é o dispositivo que executa programas que usam os protocolos SNMP e SSDP. Nos testes realizados foi utilizado um computador Windows executando o Windows Media Player e o serviço SNMP. O Windows Media Player utiliza o UPnP para o compartilhamento de mídia via DLNA[1]. O serviço SNMP [28] é a implementação da Microsoft de um agente SNMP, que é um recurso opcional do Windows. As configurações são as seguintes:

- Processador i5-660 @ 3,33GHz
- 8GB de memória RAM DDR3 @ 1333MHz
- Adaptador de rede RTL8111D Gigabit Ethernet
- Sistema Operacional Windows 10 Pro x64 Versão 1803

[1] `https://www.dlna.org/`

Alvo: O alvo apenas executa o programa Wireshark para monitorar os pacotes recebidos. As configurações são as seguintes:

- Processador i3-380M @ 2,53GHz
- 4GB de memória RAM DDR3 @ 1333MHz
- Adaptador de rede AR8152
- Sistema Operacional Windows 10 Home x64 Versão 1803

***Switch*:** O *Switch* é o componente responsável por fazer a distribuição dos pacotes. Para este papel foi usado um roteador wireless TL-WDR4300 da TP_Link.

Para realizar a medição do tráfego, foi usado o programa Wireshark no atacante, refletor e alvo.

A virtualização foi usada devido à facilidade de manter e configurar máquinas com sistema Linux. Como foi usada uma placa de rede virtual em modo *bridge* [2], a máquina do atacante atuava exatamente como se estivesse conectada a mesma rede do refletor e alvo. Desta forma, uma máquina física, com o mesmo poder de desempenho da virtual, se comportaria da mesma forma. Nos testes, o único prejuízo do uso da virtualização é uma performance menor do atacante. Isto não interfere nos resultados haja vista que a saturação dos refletores é alcançada antes da saturação do atacante, mesmo com o uso de um refletor que possui um hardware superior comparado a um dispositivo SSDP ou SNMP típico.

4.1 Análise de Desempenho

O primeiro passo dos testes é a análise de desempenho. O objetivo deste passo é encontrar fatores que limitem o envio do tráfego para o alvo. Para isso foram analisados as taxas de envio entre o atacante e o refletor e entre o refletor e o alvo, variando-se os níveis da ferramenta. Devido a saturação do computador atacante no nível 5, foram apresentados apenas os dados até o nível 6, haja vista que dados dos próximos níveis são apenas flutuações dos valores observados no nível 5. Os gráficos das análises se baseiam nos dados das Tabelas 4.1 a 4.2, para o SNMP, e das Tabelas 4.3 a 4.4, para o SSDP.

[2] `https://www.virtualbox.org/manual/ch06.html#network_bridged`

Tabela 4.1: SNMP – Fluxo de dados em bits por segundo.

Nível	**Enviados pelo Atacante**	**Recebidos pelo Refletor**	**Enviados pelo Refletor**	**Recebidos pelo Alvo**	**Amplificação**
1	639,6	647,7	394.468,2	394401,7	609,1
2	6.327,1	6.407,7	3.894.071,4	3.894.223,2	607,7
3	63.255,4	63.919,9	38.614.905,8	38.701.485,5	604,1
4	633.950,0	630.050,7	57.614.230,6	58.605.568,4	91,4
5	683.869,9	702.535,2	57.157.632,3	57.588.133,8	81,4
6	706.223,7	719.934,5	59.061.424,1	59.772.215,5	82,0

Tabela 4.2: SNMP – Fluxo de dados em pacotes por segundo.

Nível	**Enviados pelo Atacante**	**Recebidos pelo Refletor**	**Enviados pelo Refletor**	**Recebidos pelo Alvo**	**Amplificação**
1	1,0	1,0	33,4	33,4	33,1
2	10,0	10,0	330,4	330,4	33,0
3	100,1	99,9	3.195,2	3.202,1	32,0
4	1.003,1	984,5	4.825,1	4.908,7	4,9
5	1.082,1	1.097,7	4.815,0	4.852,5	4,4
6	1.117,4	1.124,9	4.924,2	4.984,7	4,4

Tabela 4.3: SSDP – Fluxo de dados em bits por segundo.

Nível	**Enviados pelo Atacante**	**Recebidos pelo Refletor**	**Enviados pelo Refletor**	**Recebidos pelo Alvo**	**Amplificação**
1	1.098,5	1.098,8	41.798,3	41.716,6	38,0
2	10.889,8	10.892,9	416.486,7	416.556,5	38,2
3	108.926,2	108.925,9	3.116.812,7	3.116.555,0	28,6
4	1.087.302,7	1.088.112,8	3.253.132,5	3.253.528,7	3,0
5	1.222.812,4	1.227.486,5	3.279.005,3	3.277.356,8	2,7
6	1.909.624,5	1.914.556,3	3.311.109,2	3.311.964,9	1,7

Tabela 4.4: SSDP – Fluxo de dados em pacotes por segundo.

Nível	**Enviados pelo Atacante**	**Recebidos pelo Refletor**	**Enviados pelo Refletor**	**Recebidos pelo Alvo**	**Amplificação**
1	1,0	1,0	10,0	10,0	9,9
2	10,0	10,0	100,0	100,1	10,0
3	100,1	100,1	748,2	748,1	7,5
4	999,4	1.000,1	776,8	776,8	0,8
5	1.123,9	1.128,2	783,7	783,3	0,7
6	1.755,2	1.759,7	793,2	793,3	0,5

4.1.1 Atacante e Refletor

A primeira parte da análise de desempenho é verificar o envio de dados do atacante para o refletor. O gráfico da Figura 4.1 mostra a taxa de bits e o gráfico da Figura 4.2 mostra a taxa de pacotes enviados pelo atacante e recebidos pelo refletor, usando o protocolo SNMP. Observando os dados, verifica-se uma saturação no atacante no nível 5. Neste nível, o atacante deveria estar com uma taxa de envio igual a 10x a do nível 4, entretanto a taxa obtida foi por volta de 1,2 Mbps, frente aos 633 kbps do nível 4.

Assim como no SNMP, observa-se uma saturação do atacante no nível 5 para o protocolo SSDP, conforme pode ser visto nas Figuras 4.3 a 4.4.

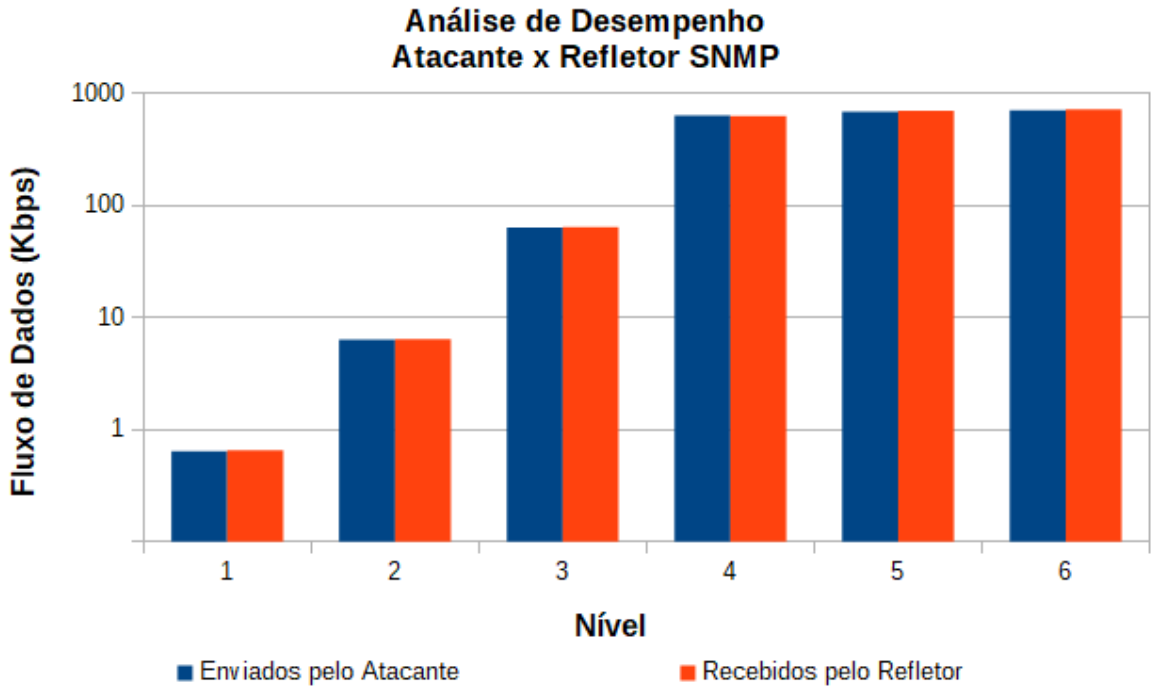

Figura 4.1: Fluxo de dados entre atacante e refletor SNMP.

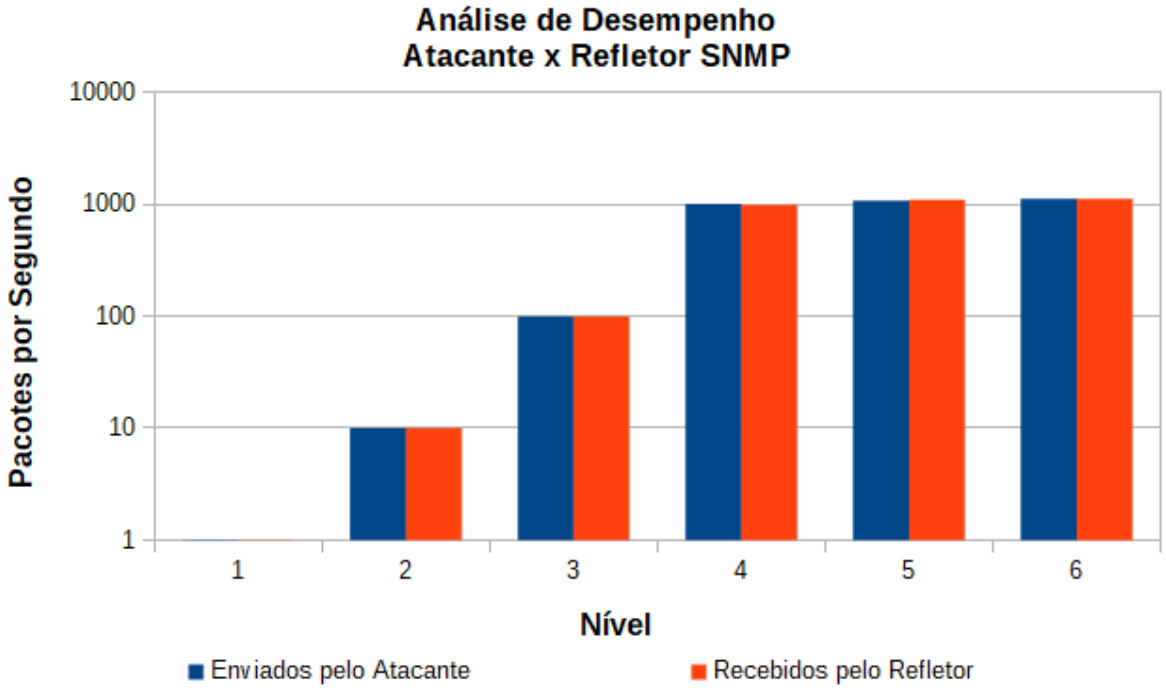

Figura 4.2: Fluxo de pacotes entre atacante e refletor SNMP.

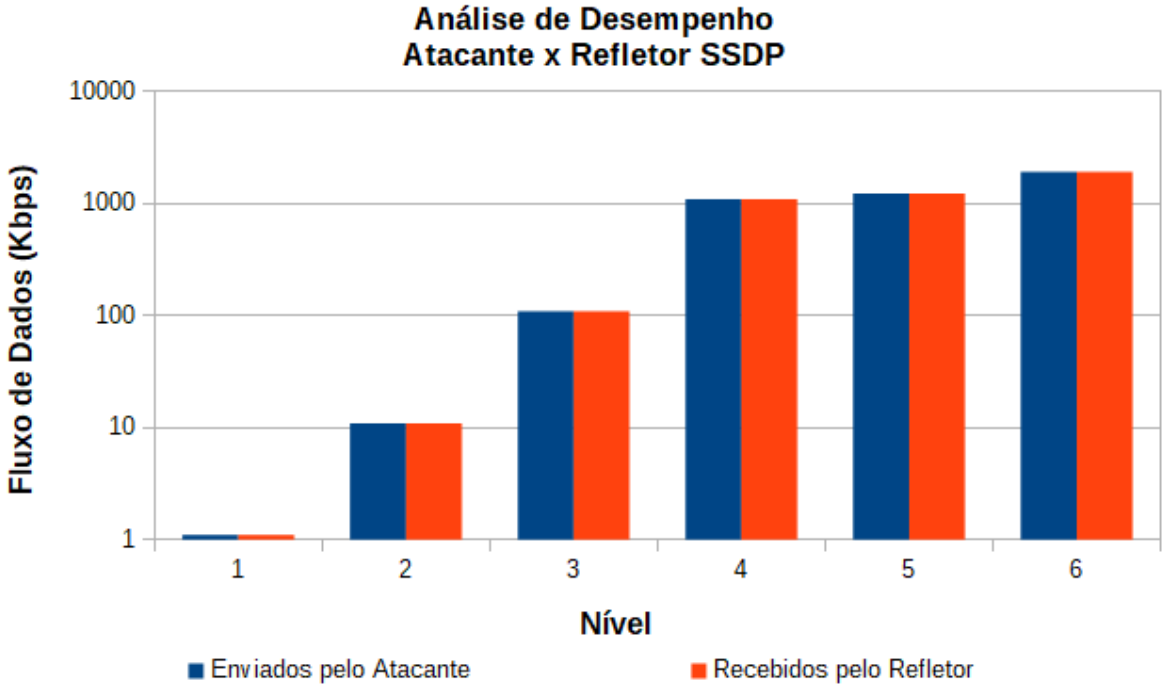

Figura 4.3: Fluxo de dados entre atacante e refletor SSDP.

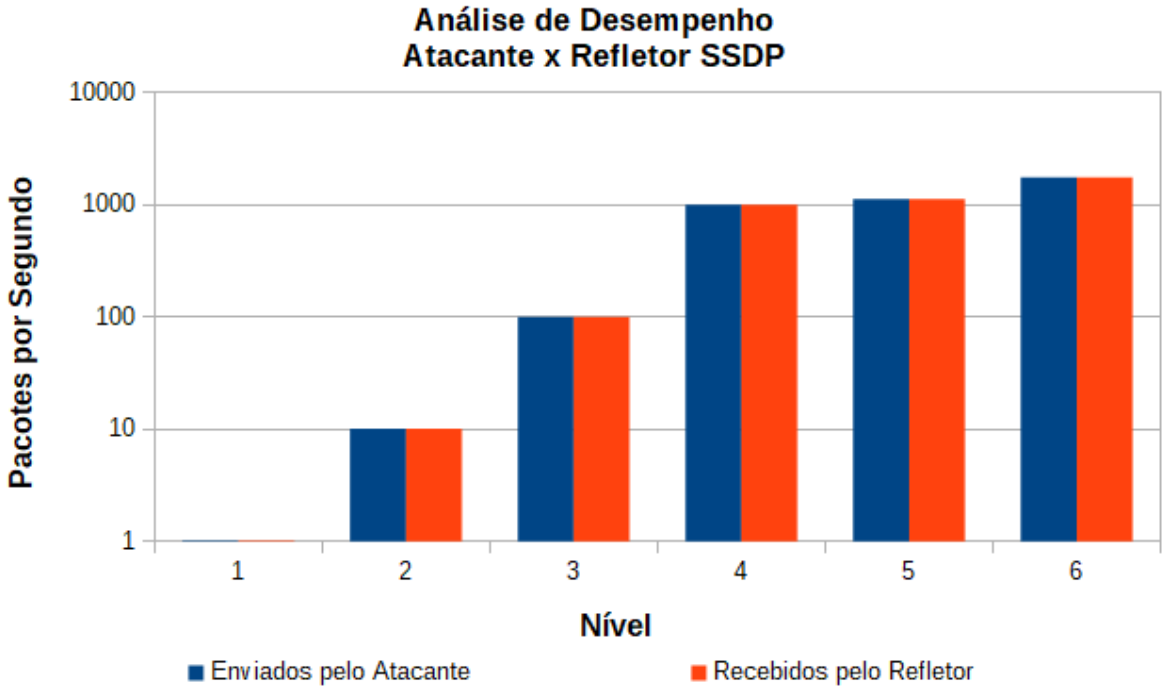

Figura 4.4: Fluxo de pacotes entre atacante e refletor SSDP.

4.1.2 Refletor e Alvo

Nesta parte busca-se encontrar gargalos entre o refletor e o alvo. O gráfico da Figura 4.5 mostra a taxa de bits e o gráfico da Figura 4.6 mostra a taxa de pacotes enviados pelo refletor e recebidos pelo alvo, usando o protocolo SNMP. Os dados mostram uma saturação do refletor no nível 4. Isto pode ser explicado devido a grande quantidade de processamento requerida para se montar a resposta a um *GetBulkRequest* com um valor de *"max-repetitons"* grande.

No caso do SSDP a saturação ocorre no nível 3, como pode ser visto nos gráficos das Figuras 4.7 a 4.8. A saturação ocorre devido ao balanceamento de carga promovido pelo uso do campo MX da mensagem M-SEARCH, conforme descrito na Subseção 2.3.2.

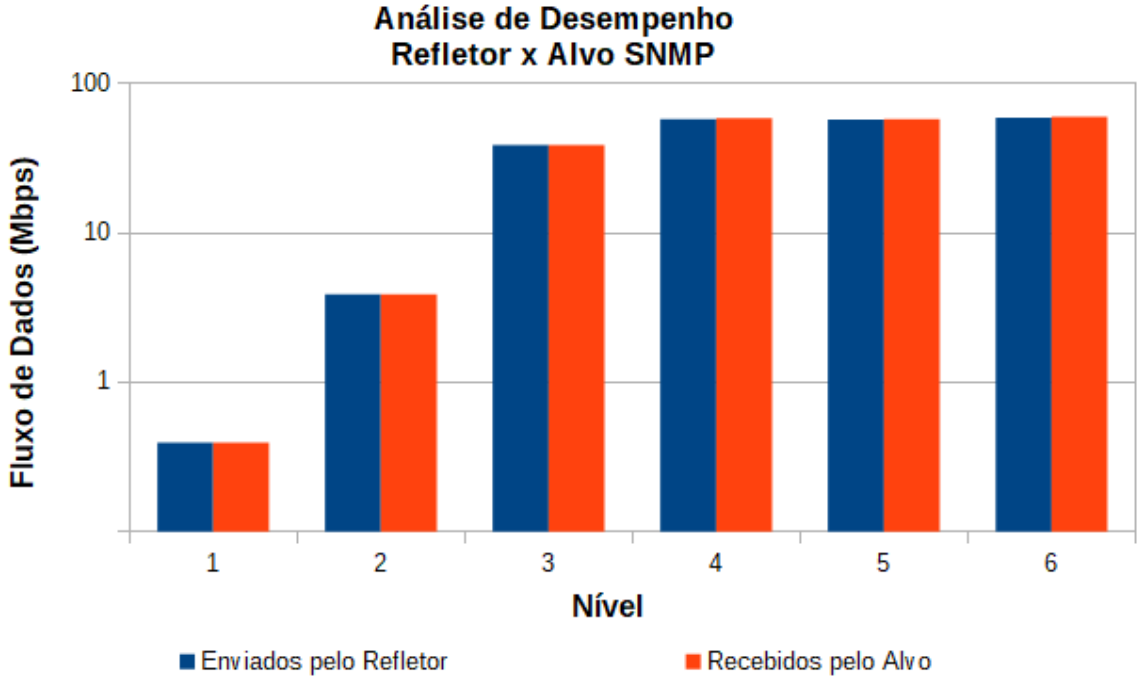

Figura 4.5: Fluxo de dados entre refletor e alvo SNMP.

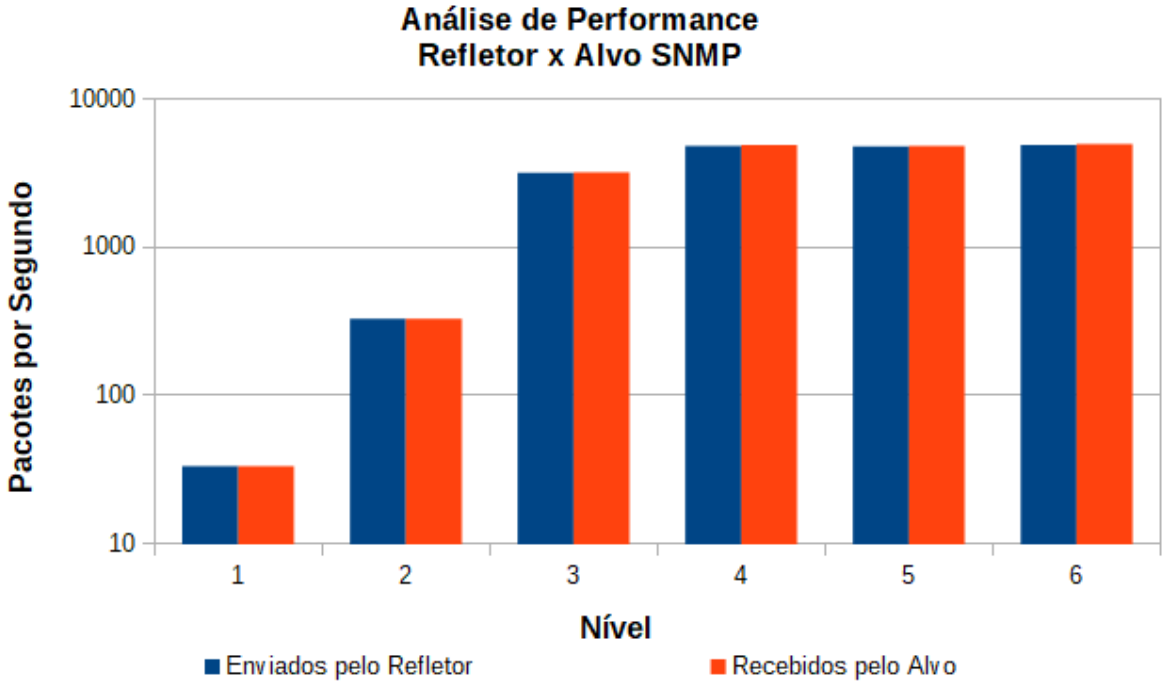

Figura 4.6: Fluxo de pacotes entre refletor e alvo SNMP.

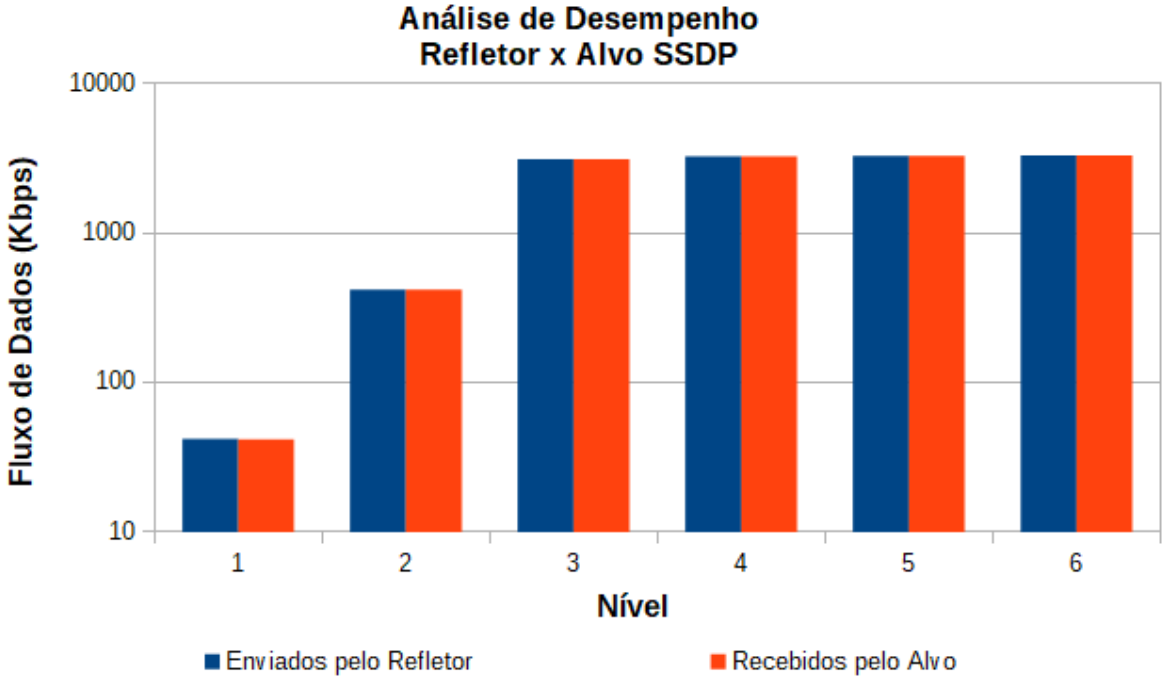

Figura 4.7: Fluxo de dados entre refletor e alvo SSDP.

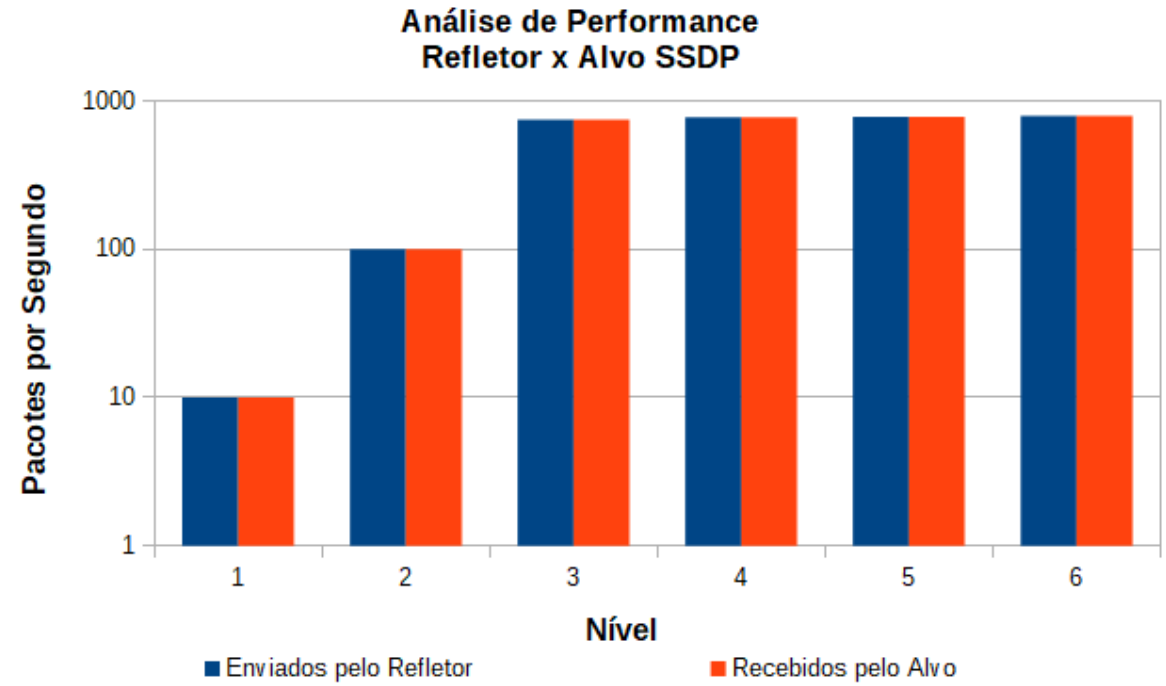

Figura 4.8: Fluxo de pacotes entre refletor e alvo SSDP.

4.2 Análise da Amplificação

Esta parte da análise consiste em verificar a amplificação máxima e a sustentabilidade desta amplificação. Conforme pode ser visto no gráfico da Figura 4.9, o ataque com SNMP conseguiu uma amplificação máxima em torno de 600 vezes, que se sustenta até o nível 3. O gráfico da Figura 4.10 mostra uma amplificação em torno de 33 vezes em relação ao número de pacotes.

Com SSDP a amplificação máxima foi em torno de 38 vezes, que se sustentou até o nível 2, conforme mostra o gráfico da Figura 4.11. Em termos de pacotes, a amplificação máxima foi em torno de 10 vezes, conforme mostra o gráfico da Figura 4.12.

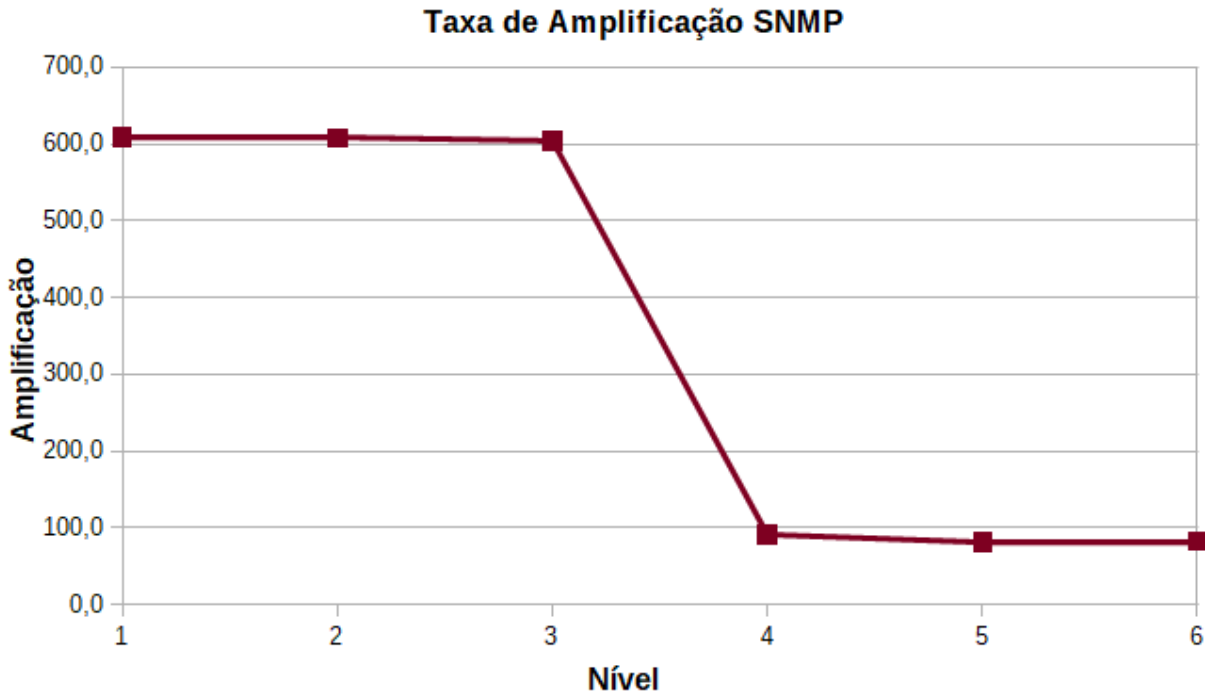

Figura 4.9: Amplificação em bits com protocolo SNMP.

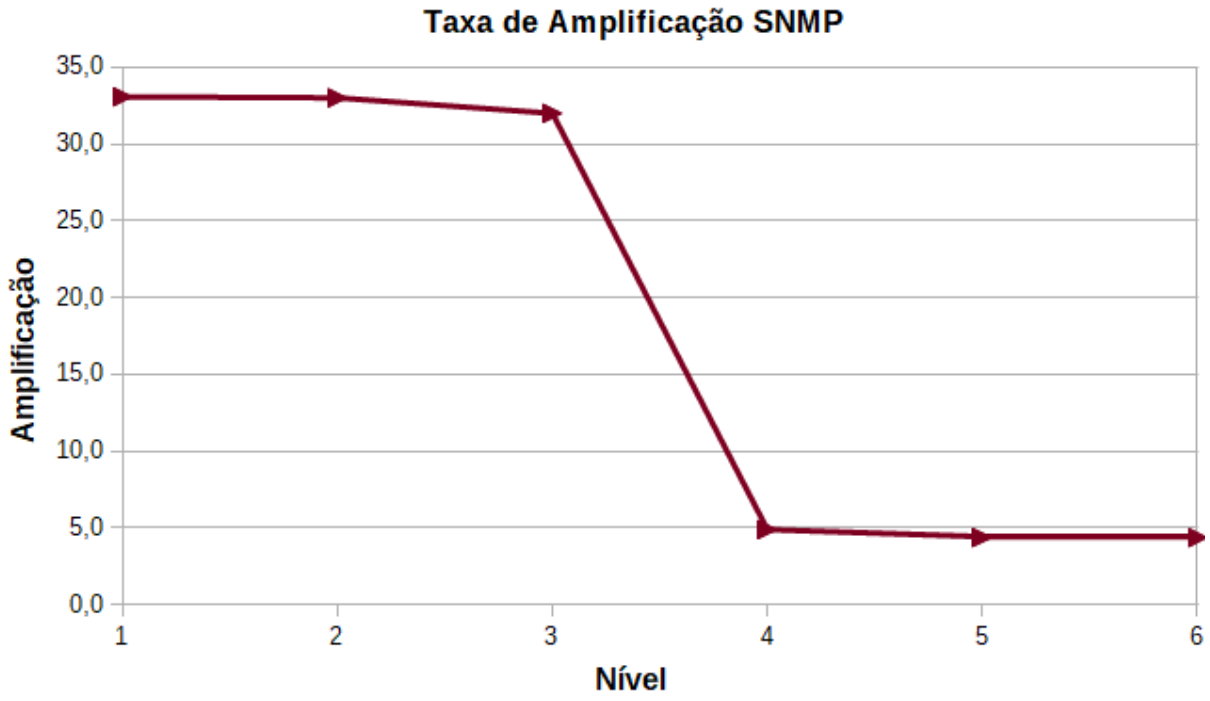

Figura 4.10: Amplificação em pacotes com protocolo SNMP.

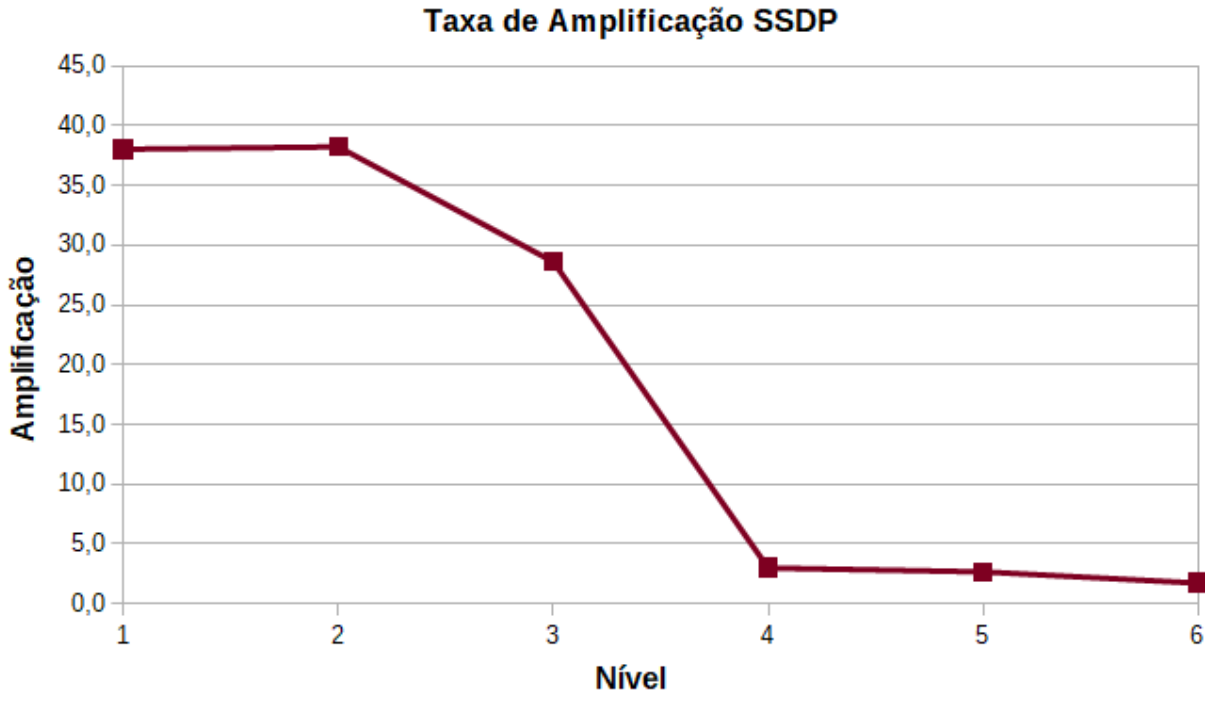

Figura 4.11: Amplificação em bits com protocolo SSDP.

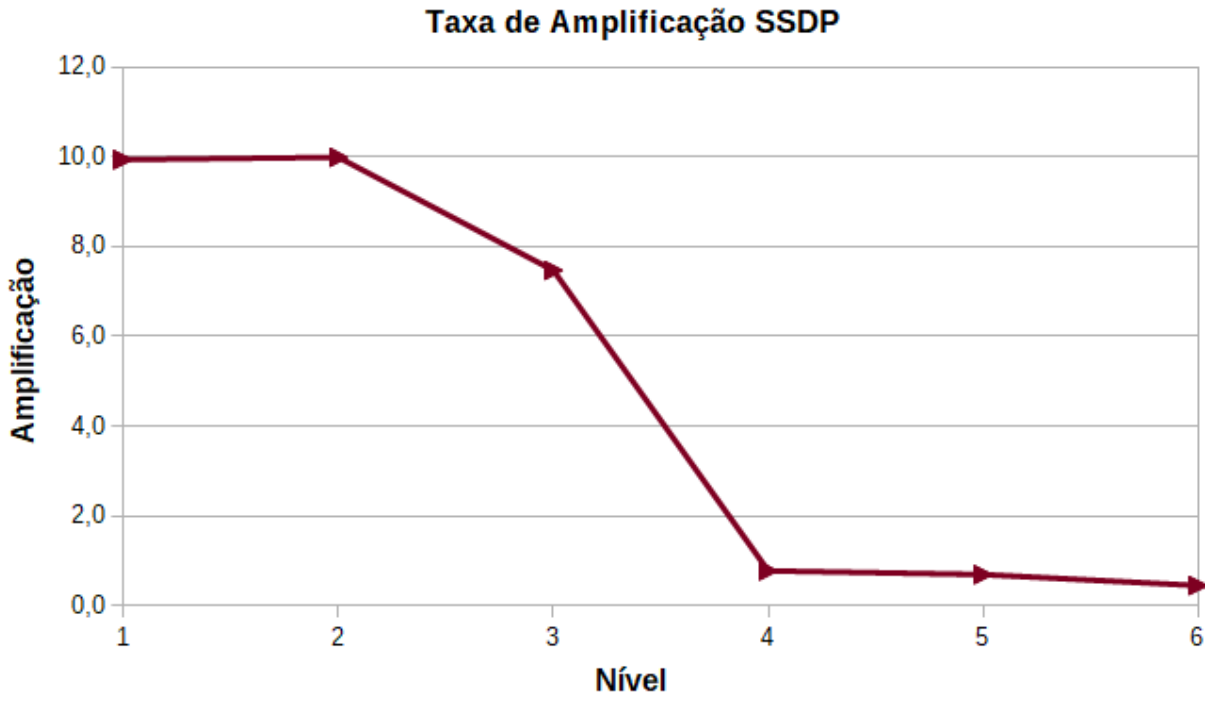

Figura 4.12: Amplificação em pacotes com protocolo SSDP.

4.3 Teste com SSDP e SNMP simultaneamente

Apenas para efeito de demonstração, foi executado um teste com os dois protocolos configurados nos níveis mais altos em que se manteve a amplificação máxima, ou seja, SSDP em nível 2 e SNMP em nível 3. Conforme pode ser visto na tabela Tabela 4.5, os bits enviados pelo atacante e os recebidos pelo alvo são aproximadamente igual a soma dos resultados obtidos nos testes com os protocolos separadamente. Isto demonstra que a ferramenta foi capaz de sustentar o ataque na situação em que há o ganho máximo com ambos os protocolos. Note que, devido a menor amplificação do SSDP em relação ao SNMP, a amplificação com ambos os protocolos acaba sendo menor que a obtida no SNMP.

Tabela 4.5: Teste com SSDP e SNMP simultaneamente.

	Enviados pelo Atacante	**Recebidos pelo Alvo**	**Amplificação**
Bits/segundo	74.275,8	40.980.790,4	540,7
Pacotes/segundo	110,3	3.536,1	31,8

4.4 Considerações Finais

Embora tenha uma amplificação significativamente menor, a adição do SSDP significa o aumento de possíveis refletores (em [9] foram achados 3.704.000 refletores em uma busca que durou uma hora), o que pode resultar em um ataque mais forte.

Os resultados obtidos com SNMP neste trabalho diferem dos obtidos em [1] devido ao uso de uma implementação SNMP diferente. Em [1] foi usado a implementação SNMP4J [29], uma API SNMP para Java, enquanto neste trabalho foi usado o Serviço SNMP do Windows [28].

Apesar de a amplificação máxima com SNMP ter se mantido até o nível 3 nos testes, o dispositivo usado não representa um refletor típico, que geralmente é um roteador. Isto significa que é possível que não se obtenha a mesma sustentação da amplificação quando o refletor possui um hardware mais fraco. Entretanto, isto não inviabiliza um ataque, já que é característica de um ataque por reflexão amplificada usar vários refletores.

Em situação de ataque real, a ferramenta pode ser utilizada por mais de um computador de ataque, de forma a tornar o ataque distribuído. Isto faria uma redução do gargalo no atacante, resultando em ataques potencialmente maiores.

Capítulo 5

Conclusão

O estudo do protocolo SSDP demonstrou sua aplicabilidade como possível amplificador em um ataque por reflexão amplificada. O método vulnerável é o M-SEARCH, que é a mensagem de busca por dispositivos UPnP. Cada requisição gera múltiplas[1] respostas de um único dispositivo, o que contribui para a amplificação de dados e pacotes.

Apesar do campo MX ser um bom método para controle de carga em uma rede privada com dispositivos confiáveis, a possibilidade de usar o valor 1, que indica que o dispositivo deve responder em um tempo aleatório entre 0 e 1 segundos, significa que um atacante pode conseguir manter uma boa amplificação.

A especificação UDA 1.0 indica que o método M-SEARCH deveria ser enviado via UDP *multicast*, o que poderia ser um entrave para o ataque. Entretanto, conforme resultados das pesquisas [9, 23] e dos próprios testes com a ferramenta, a verificação de que as requisições vem do grupo *multicast* padronizado para o SSDP não ocorre, provavelmente devido ao processamento adicional que isto requere. Com o objetivo de aumentar a abrangência, a ferramenta realiza a busca tanto por dispositivos UDA 1.0 quanto por versões mais atuais, que possuem especificação para receber mensagens de busca via *unicast*.

A viabilidade do SNMP observada por Medeiros [1] foi novamente confirmada pelos testes realizados. O diferencial do teste realizado neste trabalho foi o uso de uma implementação comercial, o que aproxima o teste do caso real.

Os testes de performance e de amplificação revelaram os seguintes resultados:

- O atacante saturou no nível 5, o que pode ser explicado devido ao gasto de processamento com *raw sockets* e ao hardware limitado do computador de ataque.

[1] 3 pelo dispositivo raiz mais 2 por dispositivo embarcado e mais 1 por serviço

- O refletor SNMP saturou nos níveis acima de 3 devido ao alto processamento gasto pelo *GetBulkRequest*. Um dispositivo típico, por exemplo um roteador, provavelmente saturaria em níveis mais baixos devido ao hardware mais fraco.
- O refletor SSDP saturou nos níveis acima de 2 devido apenas ao controle de carga, o que significa que um dispositivo com hardware mais fraco que siga as recomendações da especificação UDA provavelmente geraria o mesmo resultado.
- A fragmentação provocada pelas grandes respostas do refletor no protocolo SNMP promove uma grande amplificação de pacotes e gera um grande custo de processamento para a remontagem dos pacotes IP.

A filtragem é essencial para que apenas o tráfego legítimo chegue ao servidor hospedando o serviço. Um método eficaz é bloquear pacotes UDP com porta de origem especifico. Tanto o SNMP quanto o SSDP são protocolos que usam as portas padronizadas 161 e 1900, respectivamente. O simples bloqueio de mensagens com estas portas de origem filtraria um ataque de negação de serviço. Entretanto mesmo essa simples filtragem requer uma infraestrutura preparada para tal. As empresas então acabam por contratar empresas especializadas para realizar essa função, o que incorre em gastos adicionais. Para agravar a situação, em Maio de 2018 os pesquisadores da Imperva descobriram um jeito de ofuscar a porta de origem [30]. A técnica consiste em adicionar uma regra de redirecionamento de portas em roteadores com implementações vulneráveis do UPnP, fazendo com que o roteador atue como um *proxy* para o atacante. Caso essa ou outras técnicas descobertas futuramente sejam usadas para ofuscar a porta de origem, o único método de filtragem seria uma inspeção profunda de pacotes, o que requer uma grande infraestrutura.

Tendo em vista a viabilidade dos ataques tanto com protocolo SNMP quanto SSDP, a ferramenta se mostra útil para a detecção de dispositivos refletores e para estudos de mitigação de ataques DoS.

5.1 Trabalhos Futuros

As principais modificações feitas na ferramenta tiveram em mente a possibilidade de trabalhos posteriores. Com essas modificações, espera-se que outros trabalhos com diferentes protocolos possam ser incluídos na ferramenta, tornando-a ainda mais útil.

Um estudo detalhado de otimização pode ser feito na ferramenta de forma a torná-la mais eficiente, possibilitando ataques maiores.

Referências

[1] Medeiros, Tiago Fonseca: *Ataque distribuído de negação de serviço por reflexão amplificada usando simple network management protocol*. http://bdm.unb.br/bitstream/10483/11152/1/2015_TiagoFonsecaMedeiros.pdf, 2015. v, vi, 2, 22, 42, 43

[2] Verisign: *Verisign Distributed Denial of Service Trends Report*. http://www.verisign.com/en_US/security-services/ddos-protection/ddos-report/index.xhtml, acesso em 2018-09-27. 1

[3] Verizon: *2018 Data Breach Investigations Report*. https://www.verizonenterprise.com/resources/reports/rp_DBIR_2018_Report_en_xg.pdf, acesso em 2018-09-27. 1

[4] Kaspersky: *Ddos breach costs rise to over $2m for enterprises finds kaspersky lab report*. https://usa.kaspersky.com/about/press-releases/2018_ddos-breach-costs-rise-to-over-2m-for-enterprises-finds-kaspersky-lab-report, acesso em 2018-09-27. 1

[5] Kiss, Jemima: *Xbox live and Playstation attack: Christmas ruined for millions of gamers*. https://www.theguardian.com/technology/2014/dec/26/xbox-live-and-psn-attack-christmas-ruined-for-millions-of-gamers, acesso em 2018-12-01. 1

[6] Lewis, Dave: *The DDoS Attack Against Dyn One Year Later*. https://www.forbes.com/sites/davelewis/2017/10/23/the-ddos-attack-against-dyn-one-year-later/, acesso em 2018-12-01. 1

[7] Anthony, Sebastian: *GitHub battles "largest DDos" in site's history, targeted at anti-censorship tools*. https://arstechnica.com/information-technology/2015/03/github-battles-largest-ddos-in-sites-history-targeted-at-anti-censorship-tools/, acesso em 2018-12-01. 2

[8] Gartner: *Gartner Says 6.4 Billion Connected "Things" Will Be in Use in 2016, Up 30 Percent From 2015*. https://www.gartner.com/newsroom/id/3165317, acesso em 2018-12-03. 2

[9] Rossow, Christian: *UDP-Based Amplification Attacks*. https://christian-rossow.de/publications/amplification-ndss2014.pdf, acesso em 2018-09-27. 2, 5, 42, 43

[10] Moore, H. D.: *Security Flaws in Universal Plug and Play*. https://information.rapid7.com/rs/411-NAK-970/images/SecurityFlawsUPnP%20%281%29.pdf, acesso em 2018-09-27. 2, 8, 17

[11] Ranger, Steve: *GitHub hit with the largest DDoS attack ever seen*. https://www.zdnet.com/article/github-was-hit-with-the-largest-ddos-attack-ever-seen/, acesso em 2018-12-03. 5

[12] *Slowloris*. https://www.incapsula.com/ddos/attack-glossary/slowloris.html, acesso em 2018-11-06. 7

[13] *What is a Slowloris DDoS attack?* https://www.cloudflare.com/learning/ddos/ddos-attack-tools/slowloris/, acesso em 2018-11-06. 7

[14] *Intel® Software for UPnP Technology: Technology Overview*. https://software.intel.com/en-us/articles/intel-software-for-upnp-technology-technology-overview, acesso em 2018-11-06. 8

[15] *Portable SDK for UPnP Devices*. http://pupnp.sourceforge.net/, acesso em 2018-11-06. 8

[16] Bernard, Thomas: *MiniUPnP Project HomePage*. http://miniupnp.free.fr/, acesso em 2018-11-06. 8

[17] UPnP Forum: *UPnP Device Architecture 1.0*, Outubro 2008. http://upnp.org/specs/arch/UPnP-arch-DeviceArchitecture-v1.0.pdf. 8, 9, 11, 13, 15, 16

[18] *Internet Gateway Device (IGD) V 2.0*. https://openconnectivity.org/developer/specifications/upnp-resources/upnp/internet-gateway-device-igd-v-2-0, acesso em 2018-11-06. 9

[19] S. Cheshire, B. Aboba, E. Guttman: *Dynamic Configuration of IPv4 Link-Local Addresses*. https://tools.ietf.org/html/rfc3927, acesso em 2018-11-06. 10

[20] R. Fielding, J. Reschke: *Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing*. https://tools.ietf.org/html/rfc7230, acesso em 2018-11-06. 11

[21] *Simple Object Access Protocol (SOAP) 1.1*. https://www.w3.org/TR/2000/NOTE-SOAP-20000508/, acesso em 2018-11-07. 15

[22] H. Nielsen, P. Leach, S. Lawrence: *An HTTP Extension Framework*. https://tools.ietf.org/html/rfc2774, acesso em 2018-11-07. 15

[23] Majkowski, Marek: *Stupidly Simple DDoS Protocol (SSDP) generates 100 Gbps DDoS*. https://blog.cloudflare.com/ssdp-100gbps/, acesso em 2018-09-27. 17, 43

[24] *Gson*. https://github.com/google/gson, acesso em 2018-10-02. 23, 29

[25] Oracle: *Java Native Interface Specification*. https://docs.oracle.com/en/java/javase/11/docs/specs/jni/index.html. 26

[26] *Guava: Google Core Libraries for Java*. https://github.com/google/guava, acesso em 2018-10-02. 26

[27] *Saddam DDoS Amplification Tool*. https://github.com/OffensivePython/Saddam, acesso em 2018-12-04. 30

[28] *SNMP Service*. https://docs.microsoft.com/en-us/windows/desktop/snmp/snmp-start-page, acesso em 2018-11-12. 31, 42

[29] *The SNMP API for Java*. http://www.snmp4j.org/, acesso em 2018-11-12. 42

[30] Avishay Zawoznik, Johnathan Azaria, Igal Zeifman: *New DDoS Attack Method Demands a Fresh Approach to Amplification Assault Mitigation*. https://www.imperva.com/blog/2018/05/new-ddos-attack-method-demands-a-fresh-approach-to-amplification-assault-mitigati acesso em 2018-10-18. 44